AMSTRAD CPC 464

MANUAL DEL USUARIO

Introducción

AMSTRAD CPC464

La informática ha evolucionado mucho en muy poco tiempo. De todas las innovaciones tecnológicas de este siglo, la informática es seguramente la más notable.

Las características de los ordenadores, tanto en el aspecto mecánico y electrónico (*hardware*) como en el de la programación (*software*) avanzan tan deprisa que incluso a los usuarios expertos les resulta difícil mantenerse al día. Por consiguiente, harían falta miles de páginas para describir exhaustivamente el BASIC del CPC464, su sistema operativo y su hardware.

Este manual es solamente una introducción al CPC464 y a su software. Para complemetarlo se han publicado numerosos libros y guías de todo tipo que el usuario puede encontrar en librerías y tiendas de ordenadores.

Los usuarios que ya conozcan otros dialectos de BASIC no encontrarán ninguna dificultad en habituarse muy deprisa al AMSTRAD BASIC. Los demás apreciarán enseguida la estructura directa y clara de esta implementación del lenguaje, que ha sido diseñada para obviar las ambigüedades y el oscurantismo de otras versiones de BASIC.

El manual se divide en varios capítulos. El primero es un 'Curso de introducción', escrito para guiar al usuario en su primer contacto con el ordenador. En el segundo ofrecemos un ejemplo de concepción y desarrollo de un programa.

El tercer capítulo contiene la descripción completa y detallada de todas las órdenes y funciones de AMSTRAD BASIC.

En el capítulo cuarto, 'Para su referencia', damos diversa información técnica.

El capítulo cinco da un repaso a ciertas cuestiones básicas de la informática, particularizadas para el CPC464.

Finalmente, el apéndice I es un glosario de la terminología informática y un léxico inglés-castellano.

AMSTRAD

© Copyright 1987 AMSTRAD plc

El contenido de este manual y el producto en él descrito no pueden ser adaptados ni reproducidos, ni total ni parcialmente, salvo con el permiso escrito de AMSTRAD plc ('Amstrad').

El producto descrito en este manual, así como los diseñados para ser utilizados con él, están sujetos a desarrollo y mejoras continuas. Toda la información técnica relativa al producto y su utilización (incluida la que figura en este manual) es suministrada por Amstrad de buena fe. Admitimos, no obstante, que en este manual puede haber errores u omisiones. El usuario puede obtener una lista de correcciones y modificaciones solicitándola a Amstrad o de sus distribuidores. Rogamos a los usuarios que rellenen y envíen a los distribuidores las tarjetas de registro y de garantía.

Amstrad agradecerá el envío de comentarios y sugerencias relativos a este manual y al producto en él descrito.

Toda la correspondencia se debe dirigir a

AMSTRAD ESPAÑA
Aravaca, 22
28040 Madrid
España

Toda reparación u operación de mantenimiento de este producto debe ser confiada a los talleres autorizados por AMSTRAD ESPAÑA. Amstrad no puede aceptar ninguna responsabilidad derivada del daño o pérdida que se pueda ocasionar como resultado de reparaciones efectuadas por personal no autorizado. El objetivo de este manual no es sino servir de ayuda al usuario en la utilización del producto; por consiguiente, Amstrad queda eximido de responsabilidad por el daño o pérdida a que pueda dar lugar la utilización de la información aquí publicada o la incorrecta utilización del producto.

Z80 es marca registrada de Zilog Inc.
AMSDOS, CPC6128, CPC664, CPC472 y CPC464 son marcas registradas de
AMSTRAD plc
Edición 1987

Compilado por Ivor Spital
Escrito por Ivor Spital, Roland Perry, William Poel, Cliff Lawson;
con la colaboración de Locomotive Software Ltd
y con aportaciones de Alexander Martin, David Radisic y Ken Clark
Traducido del inglés por Emilio Benito Santos

Publicado por Amstrad

Edición española producida por
Vector Ediciones, S.A. (91-202 49 40)

AMSTRAD es marca registrada de Amstrad plc
Queda estrictamente prohibido utilizar la marca y la palabra AMSTRAD sin la debida autorización

IMPORTANTE

A lo largo de este manual vamos a utilizar la letra de tipo **Univers negra** para representar los caracteres que aparecen en la pantalla y los que el usuario debe introducir por el teclado. Los nombres de las teclas los escribiremos recuadrados; así, por ejemplo, [INTRO] indica que usted debe pulsar esta tecla, **no** escribir las letras **I**, **N**, **T**, **R** y **O**.

1. Para conectar el teclado, el monitor y la unidad modulador/fuente de alimentación a otros aparatos, siga escrupulosamente las instrucciones que damos en este manual y en los manuales de otros periféricos Amstrad. De lo contrario, el ordenador puede resultar seriamente dañado y la garantía quedará anulada.
2. No intente conectar este equipo a una red de distribución de energía eléctrica que no sea de 220-240 V c.a., 50 Hz.
3. El mantenimiento que pueda hacer el usuario no requiere en ningún caso acceder al interior de la carcasa. Así pues, no abra nunca el equipo. Confíe todas las reparaciones y operaciones de mantenimiento al servicio técnico de Amstrad España.
4. Para evitar la fatiga visual, ponga el monitor lo más alejado posible del teclado y trabaje con luz ambiente adecuada. Deje el control de brillo del monitor lo más bajo posible.
5. No obstruya los orificios de ventilación.
6. No utilice ni almacene el equipo a temperaturas demasiado altas ni demasiado bajas, ni en lugares húmedos, polvorientos o sometidos a grandes vibraciones.
7. Mantenga alejado del ordenador todo tipo de líquidos (café, refrescos, etc.). El ordenador puede resultar gravemente dañado si lo salpica con cualquier líquido. Si esto llega a ocurrir, consulte al servicio técnico de Amstrad España.
8. Cada vez que se apaga el ordenador, se pierden todos los datos y programas que pudiera haber en su memoria. Si desea grabar el programa antes de apagar el ordenador, consulte la Parte 6 del 'Curso de introducción'.
9. Evite que el magnetófono y las cintas estén sometidos a la influencia de campos magnéticos (altavoces, televisores, imanes, etc.).
10. No apague el ordenador cuando haya una cinta en el magnetófono y esté pulsado el botón **PLAY** (o **REC** y **PLAY**).
11. Aunque no apague el ordenador, no deje la cinta en el magnetófono mucho tiempo con el botón **PLAY** pulsado.
12. Recomendamos el uso de cintas especiales para informática. No obstante, las cintas de audio de buena calidad son igualmente adecuadas, a condición de que no sean de dióxido de cromo ni de 'metal' y de que su duración no sea superior a 90 minutos (C90). Para facilitar la localización de los programas, son aconsejables las cintas C12 (seis minutos por cara).

13. Las cintas grabadas por otros ordenadores o fabricadas para ellos no pueden ser leídas por el CPC464.

14. Las cassettes tienen una «pestaña» en su extremo posterior; cuando se rompe esta pestaña, la cinta queda protegida contra escritura, es decir, no se puede grabar en ella. En tal caso, el botón REC del magnetófono no puede bajar y usted no debe tratar de forzarlo. Si desea grabar una cinta así protegida, tape con cinta autoadhesiva el hueco que ha quedado al descubierto al romper la pestaña.

15. Antes de grabar un programa, haga avanzar la cinta lo suficiente como para que el tramo de cinta de material no sensible (de color blanco o traslúcido) quede bobinado y frente a la cabeza quede material sensible (de color marrón).

16. El buen cuidado de las cintas y la limpieza periódica del magnetófono son necesarios para garantizar que la grabación y lectura de los programas se realice libre de errores.

17. En el zócalo de 'UNIDAD DE DISCO' de este ordenador se puede conectar diversos periféricos (por ejemplo, la unidad de disco DDI-1, el interfaz serie RS232C, etc.). Muchos de estos periféricos se reservan una pequeña porción de la memoria del ordenador. Algunos de los primeros programas publicados para el CPC464 aprovechan esta zona de memoria, y por lo tanto no pueden funcionar cuando se conectan los periféricos citados.

Contenido

Capítulo 1
Curso se introducción

Instalación y mantenimiento 1.1
Conexión de periféricos 1.7
Familiarización con el teclado 1.9
Carga de programas 1.14
Introducción a las palabras clave de BASIC 1.20
Grabación en cinta 1.36
Modos de pantalla, colores y gráficos 1.38
Sonidos 1.54

Capítulo 2
La génesis de un programa

Un programa sencillo 2.2
Utilización de las listas 2.2
Introducción de un menú 2.5
Grabación y lectura de las variables 2.8
Numeración de las líneas 2.8

Capítulo 3
Lista completa de las palabras clave del BASIC de AMSTRAD

Descripción de la notación utilizada
Lista alfabética de las palabras clave, para las que se da:
- Palabra clave
- Sintaxis
- Descripción
- Observaciones especiales (cuando son necesarias)
- Palabras clave asociadas

Capítulo 4
Para su referencia . . .

Posiciones del cursor y códigos ampliados de control 4.2
El sistema operativo 4.6
Interrupciones 4.7

Caracteres ASCII y caracteres gráficos 4.8
Esquemas del teclado 4.21
Sonidos 4.24
Mensajes de error 4.28
Palabras reservadas de BASIC 4.31
Plantillas 4.32
Conexiones 4.36
Impresoras 4.39
Joysticks 4.40
Ampliaciones del sistema residentes (RSX) 4.41
Memoria 4.42

Capítulo 5
Cuando usted guste ...

Sobre la informática en general:
- El mundo de los ordenadores
- Hardware y software
- Comparación de unos ordenadores con otros
- Aclaración de algunos conceptos
- Cómo maneja el ordenador las instrucciones del usuario
- El mundo digital
- Bits y bytes
- Sistema binario
- Sistema hexadecimal

Sobre el CPC464 en particular:
- Juego de caracteres
- Variables
- Lógica
- Caracteres definidos por el usuario
- Formatos de escritura
- Ventanas
- Interrupciones
- Datos
- Sonido
- Gráficos

Apéndices

Apéndice 1 Léxico inglés-castellano y glosario
Apéndice 2 Algunos programas de juegos:
- Rebotes
- Bombardero
- Amthello

Apéndice 3 Índice

Capítulo 1
Curso de introducción

Parte 1: Instalación y mantenimiento

El CPC464 puede ser instalado con:

1. El monitor de fósforo verde AMSTRAD GT64.
2. El monitor de color AMSTRAD CTM640.
3. O la unidad modulador/fuente de alimentación AMSTRAD MP1 conectada a un televisor de color (en UHF).

Conexión del ordenador a la red

El CPC464 sólo se puede conectar a la red de 220-240V c.a., 50 Hz.

Desenchufe la clavija de la toma mural siempre que no esté utilizando el ordenador.

No extraiga ningún tornillo ni trate de abrir la carcasa del ordenador, del monitor ni de la unidad MP1. Lea y respete las advertencias que se dan en la etiqueta de características, que está situada en la cara inferior de la carcasa del ordenador y de la fuente de alimentación:

Desconectar cuando no se utilice.

¡ATENCIÓN!

No manipular en su interior.

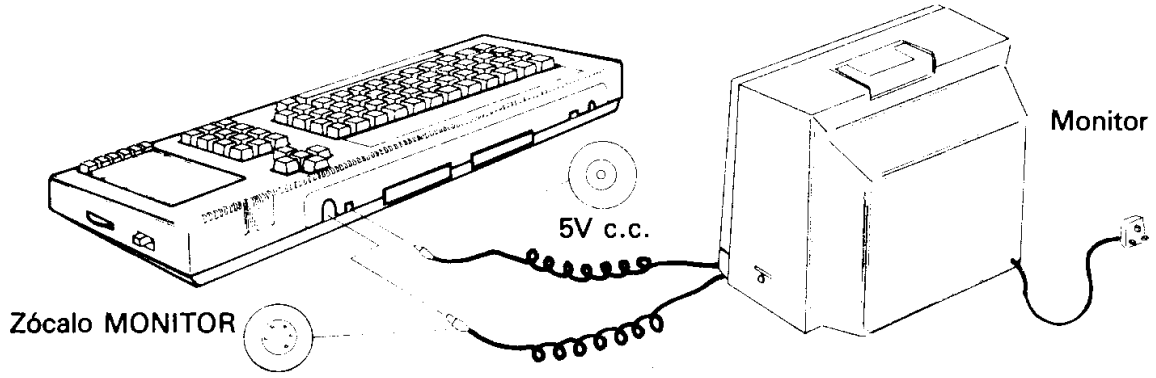
y en la cara posterior de los monitores:

NO ABRIR SIN ANTES DESCONECTAR
LA TENSIÓN DE RED.

Conexión del ordenador a un monitor

(Si va a utilizar su CPC464 con la unidad MP1, no es necesario que lea esta sección.)

1. Cerciórese de que el monitor no está conectado a la red.
2. Conecte el cable que sale de la cara frontal del monitor y termina en una clavija grande (DIN de 6 patillas) al zócalo posterior del ordenador marcado con **MONITOR.**
3. Conecte el cable que sale de la cara frontal del monitor y termina en la clavija más pequeña al zócalo posterior del ordenador marcado con **5V c.c.**

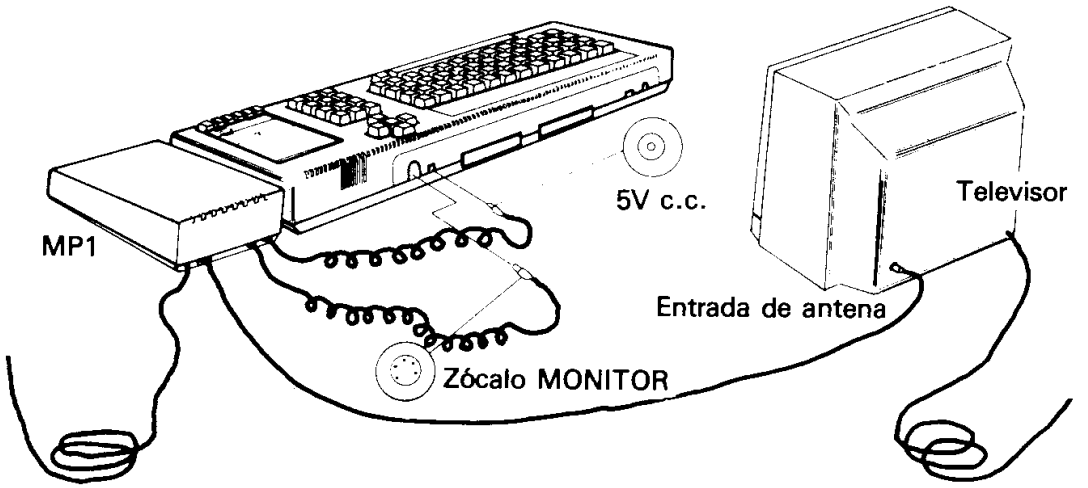


Conexión del ordenador a la unidad modulador/fuente de alimentación MP1

La MP1 es una unidad opcional que el usuario puede querer adquirir si está utilizando actualmente el CPC464 con el monitor de fósforo verde GT64. La MP1 permite conectar el CPC464 a un televisor de color doméstico, para así disfrutar de las posibilidades de color del CPC464.

La MP1 se debe colocar inmediatamente a la derecha del CPC464.

1. Cerciórese de que la MP1 no está conectada a la red.
2. Conecte el cable de la MP1 que termina en una clavija grande (DIN de 6 patillas) al zócalo posterior del ordenador marcado con **MONITOR.**
3. Conecte el cable de la MP1 que termina en una clavija pequeña al zócalo posterior del ordenador marcado con **5V c.c.**
4. Conecte el cable que sale de la MP1 y termina en una clavija de antena a la entrada de antena de su televisor.



Encendido del sistema CPC464 con GT64 o con CTM640

(Si va a utilizar su CPC464 con la unidad MP1, no es necesario que lea esta sección.)

Una vez conectado el sistema según se ha explicado en secciones anteriores, inserte la clavija de red en la toma mural. Para encender el sistema, pulse el botón **ALIMENTACION** que está en la cara frontal del monitor, de modo que quede en posición «**ENC.**». Si este botón está en «**APAG.**», el suministro de corriente al resto del sistema queda interrumpido.

Encienda el ordenador mediante el interruptor deslizante marcado con **ALIMENTACION**, que está en su cara derecha.

En este momento se debe encender el piloto rojo (**ENC.**) que está en el centro del teclado; en el monitor se verá el siguiente mensaje:

```
Amstrad 64K Microcomputer  (s1)
©1984 Amstrad Consumer Electronics plc
        and Locomotive Software Ltd.
BASIC 1.0
Ready
```

Para evitar la fatiga visual excesiva, ajuste el control de brillo marcado con **BRILLO** al mínimo necesario para que el texto se vea cómodamente, sin que deslumbre ni resulte borroso.

El control **BRILLO** se encuentra en la parte inferior de la cara frontal del monitor GT64 y en el lateral derecho del CTM640.

En el caso del GT64, puede ser necesario ajustar los controles de contraste (**CONTRASTE**) y de fijación del cuadro vertical (**SINCRO. VERT.**), que están en el panel frontal.

El mando de **CONTRASTE** se debe poner en el mínimo compatible con la cómoda visualización de los textos.

Encendido del sistema CPC464 con la unidad modulador/fuente de alimentación MP1

Una vez conectado el sistema según se ha explicado en secciones anteriores, inserte la clavija de red en la toma mural. Encienda el ordenador mediante el interruptor deslizante que está en su lateral derecho.

En este momento se debe encender el piloto rojo que está en el centro del teclado. Ahora debe sintonizar el televisor para recibir las señales del ordenador.

Si su televisor tiene un selector de canales de botonera, pulse el botón correspondiente a un canal no utilizado. Ajuste el mando de sintonía siguiendo las instrucciones del manual del televisor (si dispone de un dial de sintonía, pruebe en las proximidades del canal 36) hasta obtener la siguiente imagen:

```
Amstrad 64K Microcomputer  (s1)
©1984 Amstrad Consumer Electronics plc
        and Locomotive Software Ltd.

BASIC 1.0

Ready
```

Trate de conseguir la máxima nitidez posible. El texto aparecerá en color amarillo dorado sobre fondo azul.

Si el televisor tiene un selector de canales rotatorio, gírelo hasta que pueda ver la imagen y ésta sea estable (canal 36, aproximadamente).

Otras conexiones . . .

Si desea conectar otros periféricos, tales como

- Joystick(s)
- Impresora
- Amplificador o altavoces externos
- Dispositivos de expansión
- etcétera,

al sistema básico, consulte la Parte 2 de este «Curso de introducción».

Anomalías en el funcionamiento del sistema

Al pulsar el interruptor de alimentación no se ilumina la pantalla

— Apague el interruptor pulsándolo por segunda vez.
— Desconecte el ordenador de la toma mural.
— Enchufe en ella una lámpara que usted sepa que funciona para comprobar que llega corriente.
— Desconecte la impresora.
— Si tiene instalado un interfaz RS232, desconéctelo.
— Vuelva a enchufar el ordenador en la toma mural; si hay algún interruptor que controle el paso de corriente al enchufe, enciéndalo.
— Finalmente, vuelva a pulsar el interruptor de alimentación.

Si el problema persiste, consulte a su distribuidor.

Limpieza

Para la limpieza de la pantalla y el teclado recomendamos el uso de espumas antiestáticas en aerosol. **En ningún caso se puede utilizar productos limpiadores espiritosos.**

Parte 2: Conexión de los periféricos ...

En esta sección vamos a explicar cómo se conectan al sistema CPC464 diversos periféricos, cuyo funcionamiento se explica en las correspondientes secciones de este manual.

Joysticks (bastoncillos de mando)

El joystick AMSOFT modelo JY1 es un aparato opcional que el usuario puede desear adquirir si va a utilizar su CPC464 con programas de juegos diseñados para aprovechar las posibilidades de control y disparo del joystick.

Conecte el cable procedente del joystick en el zócalo marcado con **PUERTA DE USUARIO** en el ordenador. El CPC464 admite dos joysticks; el segundo se conecta en el zócalo que hay en la peana del primero.

En secciones posteriores de este manual daremos más información sobre los joysticks.

Impresora

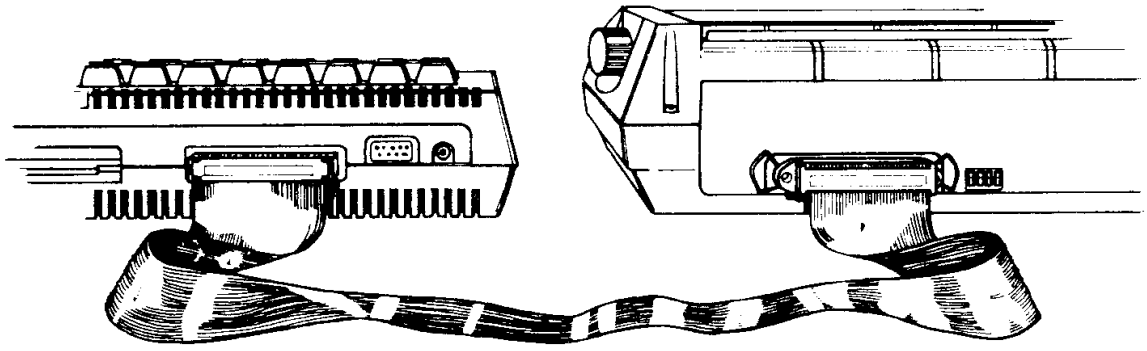
El CPC464 puede ser conectado a cualquier impresora de interfaz tipo Centronics. Para conectar el ordenador a la impresora AMSTRAD, basta con utilizar el cable suministrado con ésta.

Para conectar cualquier otra impresora de tipo Centronics se necesita el cable AMSOFT PL1.

Inserte el extremo del cable que termina en un conector plano en el zócalo del ordenador marcado con **IMPRESORA.**

Inserte el otro extremo del cable, que termina en el conector de tipo Centronics, en el zócalo de la impresora. Si la impresora tiene abrazaderas de seguridad, fíjelas en las ranuras que hay en los laterales del conector.

Más adelante daremos información sobre el manejo de la impresora.



Amplificador y altavoces externos

El CPC464 puede ser conectado a un sistema de amplificador y altavoces estereofónicos, única forma de apreciar plenamente las capacidades sonoras del ordenador.

El cable de entrada al amplificador debe teminar en una clavija estéreo de 3.5 mm, que se inserta en la hembra marcada con **E/S** en el panel posterior del ordenador.

Las conexiones de la clavija deben ser las siguientes:

Extremo de la clavija: canal izquierdo.
Anillo intermedio: canal derecho.
Cuerpo de la clavija: masa.

El CPC464 entrega a la salida **E/S** una señal de nivel constante; así pues, el volumen, el balance y el tono deben ser regulados con los mandos del amplificador externo.

También se puede conectar auriculares de alta impedancia, aunque el volumen no podrá ser regulado por el mando de **VOLUMEN** del ordenador. Los auriculares de baja impedancia, que son los habitualmente utilizados con los equipos de alta fidelidad, no pueden ser conectados directamente al ordenador.

Más adelante explicaremos cómo enviar sonidos a cada uno de los tres canales del CPC464.

Dispositivos de expansión

Al ordenador CPC464 se le pueden conectar diversos dispositivos de expansión (unidad de disco, interfaz serie, modem, lápiz fotosensible, ROMs, etc.) por medio del zócalo marcado con **UNIDAD DE DISCO** que está en la cara posterior del ordenador.

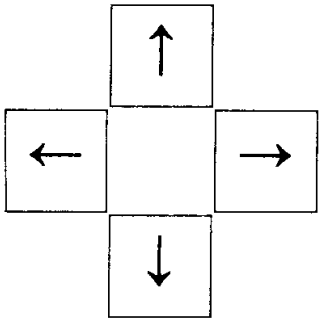
También se puede conectar en ese zócalo el amplificador/sintetizador de voz AMSOFT modelo SSA2.

Las conexiones del zócalo **UNIDAD DE DISCO** se describen en el capítulo titulado 'Para su referencia . . .'.

Parte 3: Manos al teclado ...

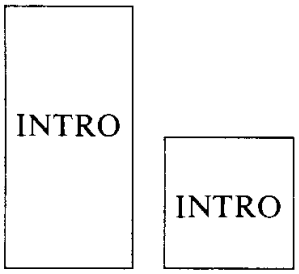
Antes de empezar a cargar y grabar programas necesitamos familiarizarnos con algunas teclas del ordenador. Si tiene usted alguna experiencia en el manejo de los ordenadores, puede omitir la lectura de esta sección.

Encienda el ordenador y verá el mensaje inicial en la pantalla. Vamos a explicar las funciones de las diversas teclas:



Las cuatro teclas que están marcadas con sendas flechas son las 'teclas de movimiento del cursor'. Estas teclas sirven, pues, para mover el cursor por la pantalla.

Púlselas todas ellas y practique hasta familiarizarse con su funcionamiento.



Hay dos teclas INTRO. Ambas sirven para introducir en el ordenador lo que usted ha tecleado. Una vez pulsada la tecla INTRO, el cursor salta automáticamente a la línea siguiente de la pantalla. Siempre que escriba una orden directa o una instrucción de programa, pulse INTRO al final.

De ahora en adelante escribiremos 'INTRO' para indicar que se debe pulsar esta tecla al terminar de teclear órdenes o instrucciones de programa.

←BORR

Esta tecla sirve para borrar el carácter (letra, número o signo) que está a la izquierda del cursor.

Teclee **abcd** y observe que el cursor queda inmediatamente a la derecha de la letra **d**. Para borrar la **d**, pulse la tecla [←BORR]. Si la mantiene pulsada durante algún tiempo, verá cómo se borran también las otras tres letras.

MAYS

Hay dos teclas [MAYS], una a cada lado del teclado. Si mantiene pulsada una de ellas al tiempo que pulsa una tecla literal (de letras), en la pantalla aparecerá la correspondiente letra *en mayúscula*.

Escriba la letra **e**, pulse la tecla [MAYS] y, antes de soltarla, vuelva a escribir la **e**. En la pantalla verá lo siguiente:

eE

Escriba ahora unos cuantos espacios (manteniendo pulsada durante algún tiempo la barra espaciadora). Para probar el efecto de [MAYS] con las teclas numéricas, escriba el **2**, pulse [MAYS] y, sin soltarla, vuelva a escribir el **2**. En la pantalla aparecerá lo siguiente:

2"

Haga pruebas con las diferentes teclas de caracteres para observar su efecto con y sin [MAYS].

FIJA MAYS

Su efecto es en cierto modo similar al de [MAYS]. Con sólo pulsarla una vez, las letras que se escriban a continuación aparecerán en mayúsculas en la pantalla, pero, en cambio, las teclas numéricas y de signos no resultan afectadas.

Pulse [FIJA MAYS] una sola vez y luego escriba lo siguiente:

abcdef123456

En la pantalla aparecerá:

ABCDEF123456

Observe que las letras han sido convertidas a mayúsculas y que, sin embargo, los números no han sido convertidos a los signos que están grabados en la parte superior de las teclas numéricas. Para obtener estos signos se pulsa la tecla correspondiente en combinación con [MAYS]. Escriba ahora lo siguiente, manteniendo pulsada la tecla [MAYS]:

abcdef123456

En la pantalla aparecerá:

ABCDEF!"#$%&

Para volver a caracteres normales (en minúsculas), pulse por segunda vez [FIJA MAYS].

Si lo que desea es obtener letras mayúsculas y los signos marcados en la parte superior de las teclas, sin tener que mantener pulsada la tecla [MAYS], puede hacer lo siguiente: mantenga pulsada la tecla [CTRL] y pulse [FIJA MAYS] una sola vez. Para comprobar el efecto de esta combinación, escriba lo siguiente:

abcdef123456

En la pantalla aparecerá:

ABCDEF!"#$%&

En esta situación, se puede escribir números utilizando el teclado numérico que está a la derecha del teclado principal.

Manteniendo pulsada la tecla [CTRL] al tiempo que se pulsa [FIJA MAYS] se vuelve al modo anterior (es decir, a minúsculas o a bloqueo de mayúsculas). Si el modo al que ha vuelto es a bloqueo de mayúsculas, pulse [FIJA MAYS] para retornar al modo normal, esto es, a minúsculas.

CLR

Esta tecla borra el carácter que está bajo el cursor.

Escriba **ABCDEFGH**. El cursor ha quedado a la derecha de la última letra (la **H**). Pulse cuatro veces la tecla [←]. El cursor se ha movido cuatro posiciones hacia la izquierda, de modo que está superpuesto a la letra **E**.

Observe que la letra **E** es visible a través del cursor. Pulse [CLR] y observe cómo desaparece la letra **E** y cómo se mueven hacia la izquierda las letras **FGH**; bajo el cursor queda la **F**. Pulse durante unos instantes [CLR]: primero desaparece la **F**, y luego la **G** y la **H**.

ESC

Esta tecla se utiliza para abandonar una función que el ordenador esté realizando. Si se pulsa [ESC] una vez, el ordenador interrumpe su tarea momentáneamente, y la reanuda si a continuación se pulsa cualquier otra tecla.

Si se pulsa [ESC] dos veces seguidas, el ordenador abandona definitivamente la tarea que está realizando y queda a la espera de otras órdenes.

Importante

Cuando se han escrito 40 caracteres en una línea, el cursor está en el extremo derecho de ella; el siguiente carácter aparecerá automáticamente al principio de la línea siguiente. Esto significa que **no se debe pulsar** [INTRO], a diferencia de lo que se haría si se estuviera trabajando con una máquina de escribir, en la que se ha de teclear el retorno del carro al acercarse al final de cada línea.

El ordenador realiza esta función automáticamente; ante un [INTRO] indebido, reaccionará con un mensaje de error, generalmente **Syntax error**, bien en el acto o bien más tarde, cuando se ejecute el programa.

Syntax error (error de sintaxis)

Cuando en la pantalla aparece el mensaje **Syntax error**, el ordenador está diciendo que no ha entendido la orden que se le ha dado.
Por ejemplo, escriba

printt [INTRO]

En la pantalla aparecerá el mensaje:

Syntax error

Esto ocurre porque el ordenador no entiende la instrucción **printt**.

Si el mismo error se comete en una línea de programa, tal como

10 printt "abc" [INTRO]

en mensaje **Syntax error** no aparece hasta que se ejecuta el programa. Escriba:

run [INTRO]

(Esta orden pide al ordenador que ejecute el programa que tiene en este momento almacenado en la memoria.) En la pantalla aparece:

Syntax error in 10
10 printt "abc"

Este mensaje indica en qué línea se ha detectado el error y exhibe la línea con el cursor ya preparado para que el usuario pueda corregirla.

Lleve el cursor, con la tecla [→], hasta una letra **t** de **printt**. Pulse [CLR] para borrar la **t** que sobra y luego pulse [INTRO] para introducir la línea corregida en el ordenador.

Escriba ahora:

run [INTRO]

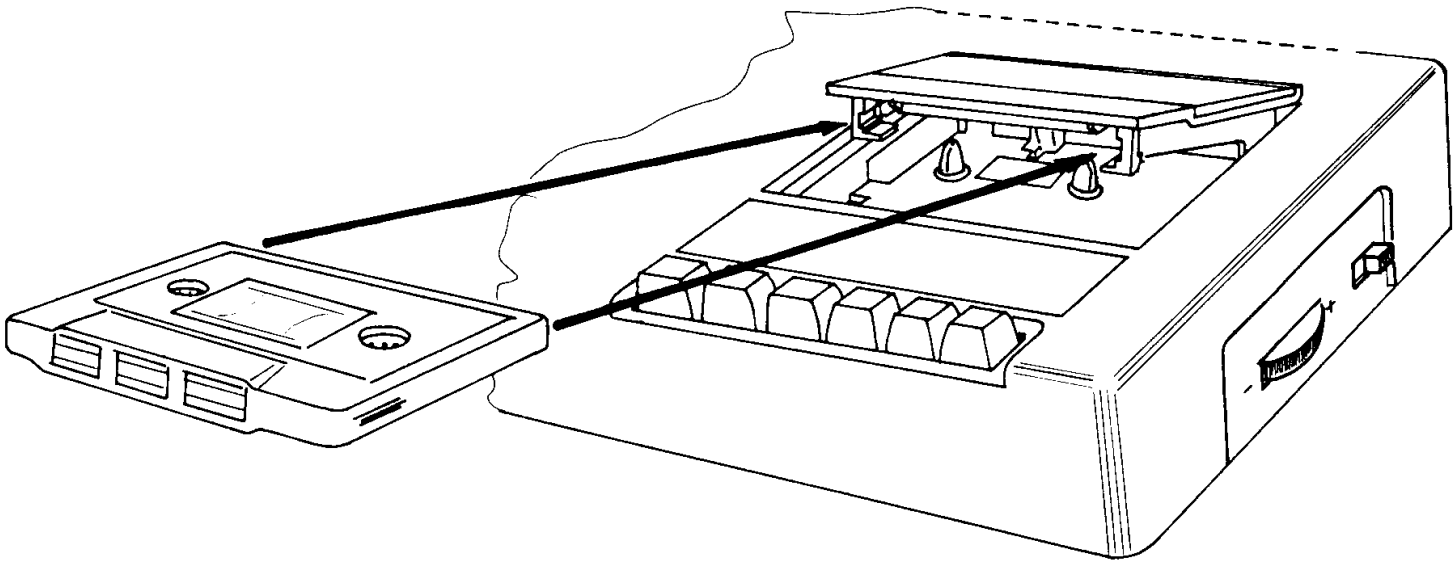
El ordenador ha entendido la instrucción; de hecho, ha escrito en la pantalla:

abc

Parte 4: Carga de programas

La cinta de bienvenida

Encajada en uno de los bloques de poliestireno del embalaje, viene una cinta de demostración. Abra la puerta del magnetófono pulsando el botón **[STOP/EJECT]**. Inserte el cassette tal como se indica en la figura, con la cara **'SIDE 1'** hacia arriba.



Cierre la puerta hasta que encaje. Pulse el botón **[REW]** para rebobinar la cinta hasta el principio (si no está ya rebobinada). Cuando se pare la cinta, pulse el botón **[STOP/EJECT]**. Ponga a 000 el contador pulsando **INICIO**.

Ahora pulse la tecla de 'control', [CTRL], y, antes de soltarla, pulse también la tecla [INTRO] pequeña que está en el extremo inferior derecho del teclado numérico. La pantalla responde de la siguiente manera:

RUN"
Press PLAY then any key:

Este último mensaje ('Pulse PLAY y luego cualquier tecla') indica que se debe pulsar el botón **[PLAY]** del magnetófono y después una tecla cualquiera en el teclado del ordenador. Por 'una tecla cualquiera' se debe entender, aquí y en lo sucesivo, cualquier tecla **menos las siguientes:** [MAYS], [FIJA MAYS], [CTRL] y [ESC].

Hecho esto, la cinta empieza a moverse. Al cabo de unos segundos aparece en la pantalla el siguiente mensaje:

Loading WELCOME 1 block 1

('Cargando WELCOME 1 bloque 1'). El proceso de carga dura unos cinco minutos. Durante este tiempo el número de bloque va cambiando a 2, 3, etc. La ejecución del programa comienza cuando se para la cinta. Observe la demostración. El programa se repite una vez y otra; para detenerlo, pulse [ESC] dos veces. Ahora ya puede pulsar el botón **[STOP]**, sacar la cinta y darle la vuelta.

Antes de cargar el programa de la segunda cara, no olvide rebobinar la cinta (botón **[REW]**).

Igual que antes, pulse [CTRL][INTRO] pequeña. En la pantalla aparece otra vez lo siguiente:

RUN"
Press PLAY then any key:

Pulse **[PLAY]** en el magnetófono y una tecla cualquiera en el teclado del ordenador.

La cinta empieza a moverse. Al cabo de unos segundos aparece en la pantalla el siguiente mensaje:

Loading WELCOME 2 block 1

('Cargando WELCOME 2 bloque 1'). El programa le invitará a participar en la demostración. Siga las instrucciones que le vaya dando.

Carga de otros programas

Estos dos programas, WELCOME 1 y WELCOME 2, son programas protegidos; esto quiere decir que la única forma de ejecutarlos es la descrita (carga y ejecución inmediata). Sin embargo, los programas de BASIC no protegidos pueden ser cargados también por los siguientes métodos.

Rebobine la cinta (**[REW]**) hasta que deje de girar y entonces pulse el botón **[STOP/EJECT]**.

Reinicialice el ordenador: para ello, pulse [CTRL] y [MAYS] y, antes de soltarlas, pulse [ESC]. La pantalla se borra y en ella aparece el mensaje de presentación inicial, exactamente igual que si se acabase de encender el ordenador.

En las siguientes instrucciones, [INTRO] significa que se debe pulsar la tecla 'INTRO' al terminar de escribirlas. El símbolo " se obtiene con [MAYS] y la tecla del **2** de la fila superior del teclado.

Escriba lo siguiente:

load "" [INTRO]

El ordenador le pide entonces que

Press PLAY then any key:

('Pulse PLAY y luego cualquier tecla'). Así pues, pulse el botón **[PLAY]** del magnetófono, ejerciendo la presión necesaria para que quede encajado. Después pulse una tecla cualquiera en el ordenador.

La cinta empieza a moverse y en la pantalla aparece poco después el siguiente mensaje:

Loading ⟨*nombre del programa*⟩ **block 1**

('Cargando ... bloque ...). El número de bloque aumenta hasta un máximo que depende de la longitud del programa. Cuando se termina de cargar el programa, el ordenador emite el mensaje:

Ready

También se puede especificar el nombre del programa que se desea cargar:

load "⟨*nombre*⟩" [INTRO]

Nuevamente el ordenador pide que

Press PLAY then any key:

('Pulse PLAY y luego cualquier tecla'). Pulse el botón **[PLAY]** del magnetófono, ejerciendo la presión necesaria para que quede encajado. Después pulse una tecla cualquiera en el ordenador.

La cinta enpieza a moverse. Si el programa elegido no está al principio, el ordenador continúa examinando la cinta hasta que lo encuentra. Por lo tanto se debe procurar escribir correctamente el nombre del programa.

Mientras el ordenador está buscando un programa en una cinta, puede ocurrir que encuentre otro distinto. En tal caso informará del hecho mediante el mensaje:

Found ⟨*otro nombre*⟩ **block 1**

('Encontrado ...') pero no lo cargará, sino que seguirá buscando hasta que encuentre el solicitado o se pulse la tecla [ESC].

Cuando encuentra el programa que está buscando, el ordenador emite el mensaje:

Loading ⟨*nombre*⟩ **block 1**

('Cargando ... bloque ...'). El número de bloque aumenta hasta que concluye el proceso de carga y en la pantalla aparece el mensaje:

Ready

Si antes de cargar el programa había algún otro en la memoria, será sustituido por el recién cargado.

Advertencia. Si al intentar cargar un programa se pulsa accidentalmente **[REC]** (además de **[PLAY]**) en el magnetófono, éste se pone en modo de grabación, no de lectura, y borra la cinta. Puesto que en esta situación el ordenador no puede encontrar el programa que está buscando, continuará borrando hasta el final de la cinta o hasta que se pulse la tecla [ESC]. Le sugerimos que proteja las cintas (después de grabarlas) rompiendo la «pestaña» de plástico que hay en el borde posterior del cassette (una para cada cara).

Una vez cargado el programa por alguno de estos dos últimos procedimientos, para ponerlo en marcha se da la orden:

run [INTRO]

También se puede ordenar la ejecución *directa* del programa:

run "" [INTRO]

El ordenador responde con el mensaje habitual:

Press PLAY then any key:

Una vez pulsado el botón **[PLAY]** del magnetófono y una tecla cualquiera del teclado, el ordenador carga el primer programa que encuentra en la cinta y lo ejecuta en cuanto termina de cargarlo. El proceso de carga se puede interrumpir en cualquier momento pulsando [ESC].

Cintas de programa pregrabadas

Las instrucciones que hemos dado en la sección anterior son aplicables a la carga y ejecución de los programas que el usuario puede adquirir en el comercio grabados en cinta.

No obstante, siempre conviene leer y seguir las instrucciones de carga que se den en el folleto que normalmente acompaña a las citadas cintas.

Errores de lectura

Si durante la carga de un programa o de un fichero de datos el ordenador emite el mensaje **Read error** ('error de lectura'), la cinta continúa funcionando y el ordenador sigue leyendo bloques, pero no los carga mientras no encuentre el bloque número 1 del programa que se le había pedido que cargase.

Cuando se produce un error de lectura lo que se debe hacer es parar la cinta ([**STOP/EJECT**]), rebobinarla ([**REW**]) y pulsar [**PLAY**] otra vez. El ordenador hará entonces un nuevo intento.

Los errores de lectura se pueden producir por diversas causas. Por ejemplo, por el deterioro accidental de la cinta o de su superficie sensible o por haber apagado el ordenador estando la cinta en el magnetófono y el botón [**PLAY**] (o [**REC**] y [**PLAY**]) pulsado.

En efecto, cuando está pulsado este botón, la cinta está presionada contra la cabeza grabadora; al apagar el ordenador y descargarse la fuente de alimentación, puede pasar un impulso eléctrico hacia la cinta a través de la cabeza. Aunque la cinta esté parada, ese impulso (lo mismo que el que se produce al encender el ordenador) puede alterar la información que está grabada en la cinta. Por otra parte, la cinta puede deformarse si se la deja demasiado tiempo oprimida entre el eje de arrastre y la rueda de caucho.

Los errores de lectura también se pueden producir si se pulsa [**PAUSE**] durante la lectura (o si se pulsó este botón durante la grabación), o si la cinta ha sido grabada en otro CPC464 en el que la cabeza grabadora estuviera mal alineada.

Finalmente, a veces se presentan errores de lectura sin causa aparente. La cinta de cassette no fue diseñada originalmente como medio de almacenamiento de datos digitales, y por lo tanto no es perfecta en esta aplicación. Hay razones físicas que limitan la velocidad a la que se puede grabar datos en la cinta. Si se intenta sobrepasar esa velocidad máxima (por ejemplo, en los métodos de duplicación masiva de cintas de programas de bajo precio), las grabaciones dejan de ser fiables.

Nota. Las cintas grabadas por o para ordenadores de otras marcas no pueden ser leídas por el CPC464. Pueden tener el mismo aspecto externo, e incluso producir los mismos ruidos cuando se las reproduce en un sistema de audio, pero los datos estarán grabados con un formato incompatible en el CPC464.

Catálago de los ficheros de la cinta

La orden **CAT** hace que el ordenador examine la cinta y dé una lista de los ficheros que encuentra en ella. La forma es la siguiente:

```
CAT [INTRO]
Press PLAY then any key:
```

La lista consiste en los nombres de los ficheros, seguido cada uno por el número de bloques de que consta y por un carácter que indica la naturaleza del fichero:

$ para programas de BASIC en forma estándar
% para programas de BASIC protegidos
***** para ficheros de texto ASCII
& para ficheros binarios

El mensaje **Ok** al final de cada línea indica que el fichero es legible y que el ordenador habría podido cargarlo si lo hubiese intentado.

La obtención del catálogo con **CAT** no afecta en absoluto al programa que pueda haber en la memoria del ordenador.

Parte 5: Empecemos a trabajar ...

A estas alturas ya sabemos qué podemos y qué no podemos hacer con el ordenador, así como la forma de conectarle periféricos. Sabemos para qué sirven algunas teclas del ordenador y cómo cargar programas. Ahora vamos a ver algunas de la órdenes e instrucciones que usted puede teclear para que empiecen a ocurrir cosas . . . .

Al ordenador le ocurre lo que a los humanos: sólo puede entender instrucciones que se le dan en el lenguaje que él conoce. En el caso del ordenador, ese lenguaje es BASIC (siglas de *Beginners' All-purpose Symbolic Instruction Code*, 'código de instrucciones simbólicas de uso general para principiantes'). Las palabras del vocabulario de BASIC son la llamadas 'palabras clave', 'palabras reservadas' o 'palabras de instrucción'. Cada una de ellas ordena a la máquina que realice una determinada función. Todos los lenguajes tienen sus reglas gramaticales, y BASIC no es la excepción. Aquí, el concepto de gramática se reduce al de 'sintaxis'; de ahí que el ordenador tenga la amabilidad de decirnos de vez en cuando que hemos cometido un error de sintaxis: **Syntax error**.

Introducción a las palabras clave del BASIC de AMSTRAD

En el tercer capítulo, titulado 'Lista completa de las palabras clave del BASIC de ASMTRAD', daremos una descripción de todas las palabras del dialecto de BASIC que entiende este ordenador. En esta sección vamos a presentar sólo las que se utilizan con mayor frecuencia.

CLS

Para borrar la pantalla escriba:

cls [INTRO]

Como puede observar, la pantalla efectivamente se borra y en su extremo superior izquierdo aparecen la palabra **Ready** y el cursor ■.

Para introducir palabras clave de BASIC valen tanto las letras mayúsculas como las minúsculas.

PRINT

Esta instrucción sirve para hacer que el ordenador escriba en la pantalla caracteres sueltos, palabras completas, frases o números. Teclee la siguiente orden:

print "hola" INTRO

En la pantalla puede ver:

hola

Las comillas " " indican al ordenador qué es lo que debe escribir. La palabra **hola** apareció en la pantalla en cuanto se pulsó INTRO. Teclee

cls INTRO

para borrar la pantalla.

RUN

En el ejemplo anterior hemos visto cómo obedece el ordenador una *orden directa*. Pero esto no siempre es deseable, ya que el ordenador olvida la orden inmediatamente después de ejecutarla. Podemos almacenar en la memoria del ordenador una sucesión de instrucciones para que más tarde sean ejecutadas en un orden determinado. Tal sucesión de instrucciones constituirá un *programa*. Las instrucciones de BASIC que podemos incluir en un programa tienen la forma que hemos visto, pero van precedidas de un *número de línea*. Si el programa consta de más de una instrucción, los números de línea indican al ordenador en qué orden debe ejecutarlas. Cuando después de teclear una línea de instrucción se pulsa INTRO, la línea queda almacenada en la memoria hasta que pidamos al ordenador que ejecute el programa. Escriba lo siguiente:

10 print "hola" INTRO

Observe que en este caso, aunque ha pulsado INTRO, la palabra **hola** no ha aparecido en la pantalla, sino que ha quedado almacenada en la memoria del ordenador, incluida en la línea de programa. Para ejecutar este pequeño programa debemos dar al ordenador la orden directa **run**. Escriba

run INTRO

Ahora sí aparece la palabra **hola** en la pantalla.

Observe algo interesante: en lugar de escribir la palabra **print** completa, basta con teclear el signo de interrogación **?**; por ejemplo,

10 ? "hola" INTRO

LIST

Cuando se tiene un programa almacenado en la memoria, se puede comprobar su contenido haciendo un 'listado'. Escriba

list [INTRO]

En la pantalla aparece

10 PRINT "hola"

que es la única línea de nuestro programa.

Observe que ahora la palabra **PRINT** está en mayúsculas. Esto nos indica que el ordenador ha reconocido **PRINT** como palabra clave de BASIC.

Escriba **cls** [INTRO] para borrar la pantalla. Observe que, aunque se ha borrado el texto que había en la pantalla, el programa sigue estando en la memoria del ordenador.

GOTO

Esta instrucción pide al ordenador que salte de la línea actual a la línea especificada, para no ejecutar un grupo de instrucciones, si el salto es hacia delante, o para formar un bucle, si el salto es hacia atrás. Escriba

10 print "hola" [INTRO]
20 goto 10 [INTRO]

y luego

run [INTRO]

Como puede ver, el ordenador escribe repetidamente la palabra **hola**, a la izquierda de la pantalla, pasando cada vez a la línea siguiente. La razón es que, al llegar a la línea 20, la instrucción **goto 10** provoca un salto a la línea 10.

Para detener el programa, pulse [ESC] una vez. Para reanudarlo, pulse cualquier otra tecla. Para detenerlo definitivamente, de forma que se pueda introducir otras instrucciones, pulse [ESC] dos veces.

Escriba ahora

cls [INTRO]

para borrar la pantalla.

Para inhibir el salto a la línea siguiente de la pantalla cada vez que el ordenador escriba **hola**, introduzca nuevamente el programa anterior, pero poniendo ahora un signo de punto y coma (;) al final de las comillas:

```
10 print "hola"; [INTRO]
20 goto 10 [INTRO]
run [INTRO]
```

El punto y coma indica al ordenador que debe escribir el siguiente grupo de caracteres inmediatamente a la derecha del anterior (suponiendo que quepan en la misma línea).

Para detener este programa pulse [ESC] dos veces. Escriba otra vez la línea 10, pero poniendo una coma (,) en lugar del punto y coma (;):

```
10 print "hola', [INTRO]
run [INTRO]
```

Como puede observar, la coma ha indicado al ordenador que escriba el siguiente grupo de caracteres 13 posiciones a la derecha del lugar en que empezó a escribir el anterior. Esta función es útil cuando se quiere visualizar información en columnas. No obstante, si el número de caracteres del grupo es mayor que 12, la escritura empezará otras 13 posiciones más a la derecha, de forma que entre columnas siempre quedará algún espacio.

Este número, 13, es modificable con la instrucción **ZONE**, que describiremos más adelante.

Para detener este programa, pulse [ESC] dos veces. Para borrar completamente la memoria del ordenador, pulse las teclas [CTRL] y [MAYS] y, antes de soltarlas, pulse [ESC].

INPUT

Esta instrucción hace que el ordenador quede a la espera de que el usuario introduzca información por el teclado; por ejemplo, en respuesta a alguna pregunta.

Escriba lo siguiente:

```
10 input "cuantos años tienes";edad [INTRO]
20 print "pues no aparentas tener";edad;"años" [INTRO]
run [INTRO]
```

En la pantalla aparece la pregunta:

```
cuantos años tienes?
```

Escriba su edad y pulse [INTRO]. Si, por ejemplo, su edad es 18 años, el programa escribe lo siguiente:

```
pues no aparentas tener 18 años
```

Este ejemplo ilustra la utilización de **input** combinada con una variable numérica. Hemos puesto la palabra **edad** al final de la línea 10, y por consiguiente en la memoria, para que el ordenador pueda asociarla al número que el usuario introduzca; después, en la línea 20, en lugar de **edad** escribirá ese número. Nada nos obligaba a elegir precisamente ese nombre, **edad**, y podíamos haber utilizado una letra cualquiera, por ejemplo, **b**.

Reinicialice el ordenador para borrar la memoria (teclas [CTRL] [MAYS] [ESC]). Si queremos captar por el teclado una respuesta que esté formada por caracteres cualesquiera (letras solas o letras mezcladas con números), debemos poner el signo de dólar (**$**) al final del nombre de la variable. Las variables de este tipo son las que denominamos *variables literales.*

Escriba el siguiente programa (observe que en la línea 20 hay que poner un espacio después de la primera coma, pero no antes de la segunda):

```
10 input "como te llamas";nombre$ [INTRO]
20 print "hola, ";nombre$;", yo me llamo Rolando" [INTRO]
run [INTRO]
```

En la pantalla puede ver:

```
como te llamas?
```

Escriba su nombre y luego pulse [INTRO]. Si el nombre introducido es, por ejemplo, **Manolo**, el programa escribe:

```
hola, Manolo, yo me llamo Rolando
```

En este caso hemos usado **nombre$** como nombre de la variable literal, pero perfectamente podríamos haber elegido una letra cualquiera, por ejemplo, **a$**. Vamos a combinar los dos ejemplos anteriores en un solo programa.

Reinicialice la máquina con [CTRL] [MAYS] [ESC]. Escriba lo siguiente:

```
5  cls [INTRO]
10 input "como te llamas";a$ [INTRO]
20 input "cuantos años tienes";b [INTRO]
30 print "desde luego, ";a$;", no aparentas tener";b;"años" [INTRO]
run [INTRO]
```

En este programa hemos utilizado dos variables: **a$** para representar el nombre y **b** para representar la edad. En la pantalla aparece la primera pregunta:

```
como te llamas?
```

Escriba su nombre (supongamos que es **Manolo**) y luego pulse [INTRO]. El ordenador le pregunta ahora:

```
cuantos años tienes?
```

Escriba su edad (supongamos que es **18** años) y luego pulse INTRO.

Si lo que hemos supuesto es cierto, en la pantalla aparecerá la siguiente frase:

desde luego, Manolo, no aparentas tener 18 años

Edición de programas

Si alguna de las líneas del programa anterior hubiera sido mecanografiada incorrectamente, lo que podría provocar el mensaje de error **Syntax error** o algún otro, en lugar de escribirla de nuevo es posible corregirla ('editarla' es como se dice en informática). Para ilustrar cómo hacerlo, vamos a escribir incorrectamente el programa:

```
5 clss INTRO
10 input "como t llamas";a$ INTRO
20 input "cuantos años tienes";b INTRO
30 print "desde luego,";a$;", no aparentas tener";b;"años" INTRO
```

En este programa hay tres errores:

En la línea 5 hemos escrito **clss** en lugar de **cls**.
En la línea 10 hemos escrito **t** en lugar de **te**.
En la línea 30 hemos olvidado poner un espacio entre la coma y el signo de cerrar comillas.

Hay tres métodos para corregir un programa. El primero consiste en escribir nuevamente la línea errónea. Al hacerlo, la línea nueva sustituye a la que había en la memoria con el mismo número.

El segundo es el 'método del cursor de edición'.

El tercero es el 'método del cursor de copia'.

Método del cursor de edición

Vamos a corregir el error de la línea 5.

Escriba lo siguiente:

edit 5 INTRO

En la pantalla aparece la línea 5, por debajo de la 30, con el cursor situado sobre la **c** de **clss**.

Para suprimir la **s** que sobra, pulse la tecla de movimiento del cursor [→] hasta que éste quede sobre la última **s** de **clss**; pulse la tecla [CLR] y observe cómo desaparece la **s**.

A continuación pulse [INTRO] para introducir la versión corregida de la línea 5 en la memoria. Escriba

list [INTRO]

y comprobará que la línea ha quedado corregida.

Método del cursor de copia

El cursor de copia es un segundo cursor, distinto del que puede ver en este momento en la pantalla, que aparece cuando se pulsa [MAYS] en combinación con alguna de las teclas de movimiento del cursor. De esta forma, el cursor de copia se separa del cursor ordinario y puede ser llevado a cualquier lugar de la pantalla.

Para corregir los errores de las líneas 10 y 30, pulse la tecla [MAYS] y, sin soltarla, pulse [↑] hasta llevar el cursor de copia al principio de la línea 10. Observe que el cursor principal no se ha movido, por lo que hay dos cursores en la pantalla. Ahora pulse varias veces [COPIA] hasta que el cursor de copia esté sobre el espacio que hay entre la **t** y la palabra **llamas**. Observe que el principio de la línea 10 ha sido copiado en la última línea de la pantalla y que el cursor principal se ha detenido en la misma columna que el de copia. Escriba la letra **e**, que aparecerá solamente en la línea inferior.

El cursor ordinario se ha movido, pero el de copia se ha quedado donde estaba. Pulse ahora [COPIA] hasta completar la copia de la línea 10. Pulse [INTRO] para transferir esta versión de la línea 10 a la memoria. El cursor de copia desaparece y el cursor ordinario queda por debajo de la nueva línea 10.

Para corregir el otro error, mantenga pulsada la tecla [MAYS] y lleve el cursor de copia con la tecla [↑] hasta el principio de la línea 30. Pulse [COPIA] hasta que el cursor de copia quede sobre las comillas que siguen a la coma. Pulse la barra espaciadora una vez, con lo que se inserta un espacio en la línea nueva. Pulse [COPIA] y no la suelte hasta que haya terminado de copiar el resto de la línea 30; pulse entonces [INTRO].

Compruebe que el programa ha quedado correctamente almacenado en la memoria escribiendo:

list [INTRO]

Nota. Para llevar el cursor (durante el proceso de edición) al principio o al final de una línea, pulse la tecla [←] o la [→] una vez manteniendo pulsada al mismo tiempo la tecla [CTRL].

Reinicialice el ordenador con [CTRL] [MAYS] [ESC].

IF

Las palabras clave **IF** (si) y **THEN** (entonces) se combinan para hacer que el ordenador realice una determinada acción en función del resultado de una comprobación especificada. Por ejemplo, en la instrucción

```
if 1+1=2 then print "correcto" [INTRO]
```

el ordenador comprueba si es cierto que 1+1=2 y obra en consecuencia.

La palabra clave **ELSE** (si no) da un segundo curso de acción para el caso de que la comprobación dé como resultado 'falso' (o sea, que la condición no se cumpla). Por ejemplo,

```
if 1+1=0 then print "correcto" else print "falso" [INTRO]
```

Vamos a ampliar nuestro programa anterior con la instrucción **IF ... THEN**. Escriba lo siguiente:

```
5 cls [INTRO]
10 input "como te llamas";a$ [INTRO]
20 input "cuantos años tienes";edad [INTRO]
30 if edad < 13 then 60 [INTRO]
40 if edad < 20 then 70 [INTRO]
50 if edad > 19 then 80 [INTRO]
60 print "Bueno, ";a$;", todavia no eres un adolescente a los";edad;
   "años":end [INTRO]
70 print "Bueno, ";a$;", a tus";edad;"años eres un adolescente":end [INTRO]
80 print "Que le vamos a hacer, ";a$;", ya no eres un adolescente a
   tus";edad;"años" [INTRO]
```

(Observe que hemos introducido dos símbolos nuevos: <, que significa 'menor que', y >, que significa 'mayor que'.) Para comprobar que el programa ha quedado correctamente introducido en la memoria escriba

```
list [INTRO]
```

y a continuación

```
run [INTRO]
```

Responda a las preguntas que le haga el ordenador y vea qué ocurre.

En este ejemplo puede observar el efecto de la instrucción **IF ... THEN**. También hemos introducido una palabra clave nueva: **END** (fin); su efecto es concluir la ejecución del programa. Si no estuviera **END** en la línea 60, el programa continuaría en la línea 70. Por lo mismo, si no hubiéramos puesto **END** en la línea 70, el programa no terminaría en ella, sino que ejecutaría también la 80. El signo de dos puntos (:) que precede a la palabra **END**

separa esta palabra de la instrucción anterior. Este signo se puede utilizar para separar instrucciones cuando interese poner varias en una misma línea de programa. También hemos incluido la línea 5, en la que borramos la pantalla. En lo sucesivo seguiremos haciéndolo, para obtener programas más 'elegantes' y evitar confusiones.

Reinicialice la máquina pulsando las teclas [CTRL] [MAYS] [ESC].

FOR ... NEXT

Estas palabras clave se utilizan cuando se desea que una parte del programa se repita cierto número de veces. Las instrucciones que deban repetirse son las que se incluyen dentro del bucle **FOR ... NEXT** (para ... siguiente).

Escriba lo siguiente:

```
5 cls [INTRO]
10 for a=1 to 10 [INTRO]
20 print "operacion numero:";a [INTRO]
30 next a [INTRO]
run [INTRO]
```

Como puede ver, la acción de la línea 20 ha sido realizada 10 veces, tal como exige la instrucción **FOR** de la línea 10. Observe también cómo se ha ido incrementando de 1 en 1 la variable **a**.

La palabra clave **STEP** (paso) se puede incluir en la instrucción **FOR ... NEXT** para especificar la forma en que debe incrementarse o decrementarse la variable de control del bucle. Por ejemplo, modifique la línea 10 y ejecute el programa:

```
10 for a=10 to 50 step 5 [INTRO]
```

El paso también puede ser negativo. Por ejemplo:

```
10 for a=100 to 0 step -10 [INTRO]
run [INTRO]
```

REM

REM es abreviatura de *REMark* (observación, comentario). Esta instrucción indica al ordenador que debe ignorar todo lo que haya después de ella en la línea de programa. Sirve, pues, para hacer anotaciones al programa; por ejemplo, el título, el significado de una variable, etc.:

```
10 REM Fulminar los invasores [INTRO]
20 S=5:REM numero de supervivientes [INTRO]
```

El signo de comilla ' (que se obtiene de la tecla del 7 con [MAYS]) es a su vez abreviatura de **:REM**. Por ejemplo,

```
10 'Fulminar los invasores [INTRO]
20 S=5 'numero de supervivientes [INTRO]
```

GOSUB

Si en un programa hay un grupo de instrucciones que deban ser ejecutadas varias veces, no es necesario escribirlas reiteradamente cada vez que el programa las necesite. Lo que se hace es ponerlas en una 'subrutina', la cual puede ser invocada siempre que se desee mediante la instrucción **GOSUB** (ir a subrutina) seguida del número de la línea donde empieza la subrutina. El final de la subrutina se señala con la instrucción **RETURN** (retorno). Cuando el programa llega a **RETURN**, la ejecución salta a la instrucción siguiente al **GOSUB** que invocó la subrutina.

(Los dos programas siguientes no hacen más que escribir la letra de una canción popular en la pantalla, y por lo tanto no es necesario que se tome la molestia de introducirlos en el ordenador. Los hemos incluido aquí solamente para demostrar cómo se pueden utilizar las subrutinas para realizar tareas repetidas.)

En el siguiente programa:

```
10 MODE 2 [INTRO]
20 PRINT "Dicen que no la quieres y la regalas" [INTRO]
30 PRINT "Peritas de Don Guindo y uvas tempranas" [INTRO]
40 PRINT "Al tribulete" [INTRO]
50 PRINT "Que no quiere bailar con usted" [INTRO]
60 PRINT "Dejala, dejala sola" [INTRO]
70 PRINT [INTRO]
80 PRINT "Es tu cara lo mismo que luna blanca" [INTRO]
90 PRINT "Y tus ojos luceros que la acompañan" [INTRO]
100 PRINT "Al tribulete" [INTRO]
110 PRINT "Que no quiere bailar con usted" [INTRO]
120 PRINT "Dejala, dejala sola" [INTRO]
130 PRINT [INTRO]
140 PRINT "Tengo yo comparado, niña, tu rostro" [INTRO]
150 PRINT "Con la luna de enero y el sol de agosto" [INTRO]
180 PRINT "Al tribulete" [INTRO]
190 PRINT "Que no quiere bailar con usted" [INTRO]
200 PRINT "Dejala, dejala sola" [INTRO]
210 PRINT [INTRO]
run [INTRO]
```

puede observar que hemos repetido varias instrucciones. Por ejemplo, la línea 40 (principio del estribillo) está repetida en la 100 y en la 180. Pongamos todo el estribillo en una subrutina, con **RETURN** al final; entonces podremos 'llamarlo' con la instrucción **GOSUB 180** cada vez que lo necesitemos. El programa queda de la siguiente forma:

```
10 MODE 2 [INTRO]
20 PRINT "Dicen que no la quieres y la regalas" [INTRO]
30 PRINT "Peritas de Don Guindo y uvas tempranas" [INTRO]
40 GOSUB 180 [INTRO]
80 PRINT "Es tu cara lo mismo que luna blanca" [INTRO]
90 PRINT "Y tus ojos luceros que la acompañan" [INTRO]
100 GOSUB 180 [INTRO]
140 PRINT "Tengo yo comparado, niña, tu rostro" [INTRO]
150 PRINT "Con la luna de enero y el sol de agosto" [INTRO]
170 END [INTRO]
180 PRINT "Al tribulete" [INTRO]
190 PRINT "Que no quiere bailar con usted" [INTRO]
200 PRINT "Dejala, dejala sola" [INTRO]
210 PRINT [INTRO]
220 RETURN [INTRO]
run [INTRO]
```

El ahorro de trabajo es evidente. El correcto diseño de subrutinas es una faceta esencial de la informática. Con ellas se puede escribir programas 'estructurados'; son la base para la adquisición de buenos hábitos de programación.

Cuando escriba subrutinas, tenga en cuenta que no está obligado a saltar siempre a su primera línea. Por ejemplo, si hay una rutina en las líneas 500 a 800, podría interesar llamarla con **GOSUB 500**, con **GOSUB 640** o con **GOSUB 790**, según las funciones de la subrutina que se desee ejecutar.

Observe que en el programa anterior hemos incluido la instrucción **END** en la línea 170 para aislar la subrutina del resto del programa. De no haberlo hecho, el programa pasaría de la línea 160 a la 180.

Aritmética sencilla

El ordenador se puede utilizar como calculadora.

Para aprender cómo hacerlo, pruebe los siguientes ejemplos. En esta sección utilizamos el signo de interrogación **?** como abreviatura de **print**. El ordenador escribe la respuesta en cuanto se pulsa [INTRO].

Suma

Escriba

```
?3+3 INTRO
 6
```

(No se teclea el signo =).

Escriba:

```
?8+4 INTRO
 12
```

Resta

(no confunda el signo menos '–' con el de subrayado '_').

Escriba:

```
?4-3 INTRO
 1
```

Escriba:

```
?8-4 INTRO
 4
```

Multiplicación

(el signo de multiplicación es el asterisco *).

Escriba:

```
?3*3 INTRO
 9
```

Escriba:

```
?8*4 INTRO
 32
```

División

(el signo de dividir es la barra inclinada hacia la derecha, /).

Escriba:

```
?3/3 INTRO
 1
```

Escriba:

```
?8/4 INTRO
 2
```

División entera

(es la división en la que se ignora el resto; su signo es la barra inclinada hacia la izquierda \).

Escriba:

```
?10\6 INTRO
 1
```

Escriba:

```
?20\3 INTRO
 6
```

Módulo

(la operación **MOD** da el resto de la división entera).

Escriba:

```
?10 MOD 4 INTRO
 2
```

Escriba:

```
?9 MOD 3 INTRO
 0
```

Raíz cuadrada

Para hallar la raíz cuadrada de un número se utiliza la función **sqr( )**. El número del que se va a extraer la raíz (esto es, el radicando) se pone entre los paréntesis.

Escriba:

```
?sqr(16) [INTRO]
 4
```

(esto significa $\sqrt{16}$).

Escriba:

```
?sqr(100) [INTRO]
 10
```

Potenciación

(en los ordenadores el signo de 'elevar a una potencia' es la flecha hacia arriba, que en el AMSTRAD se obtiene de la tecla de Pt sin [MAYS]).

La potenciación es la operación de elevar un número a una potencia; es decir, multiplicarlo por sí mismo tantas veces como indica el exponente. Por ejemplo, 3 al cuadrado (3^2), 3 al cubo (3^3), 3 a la cuarta (3^4), etc.

Escriba:

```
?3↑3 [INTRO]
 27
```

(esto significa 3^3).

Escriba:

```
?8↑4 [INTRO]
 4096
```

(esto significa 8^4).

Raíz cúbica

Extraer la raíz cúbica de un número es lo mismo que elevarlo a la potencia $\frac{1}{3}$.

Así, para hallar la raíz cúbica de 27 ($\sqrt[3]{27}$) escriba:

```
27↑(1/3) [INTRO]
 3
```

Para hallar la raíz cúbica de 125 escriba:

```
?125↑(1/3) INTRO
 5
```

Operaciones combinadas (+, −, *, /)

El ordenador entiende correctamente las operaciones combinadas, pero es necesario saber con qué orden de prioridad las realiza.

La primera prioridad es para la multiplicación y la división; la segunda, para la suma y la resta. Este orden de prioridad es válido solamente para los cálculos en los que no intervienen más que estas operaciones.

Si la operación fuera:

3+7−2*7/4

se podría pensar que el ordenador la realizaría de la forma:

3+7−2 *7 /4
=8 *7 /4
=56/4
=14

Sin embargo, el proceso de cálculo es como sigue:

3+7−2*7/4
=3+7−14/4
=3+7−3.5
=10−3.5
=6.5

Compruébelo escribiendo

```
?3+7−2*7/4 INTRO
 6.5
```

Para alterar este orden se pueden incluir paréntesis según convenga. El ordenador realiza las operaciones internas a los paréntesis antes que las externas. Compruébelo escribiendo lo siguiente:

```
?(3+7−2)*7/4 INTRO
 14
```

El orden completo de prioridad de las operaciones matemáticas es el siguiente:

↑	Potenciación
MOD	Módulo
–	Cambio de signo
* y /	Multiplicación y división
\	División entera
+ y –	Suma y resta

Más exponentes

Cuando en los cálculos van a intervenir números muy grandes o muy pequeños, es conveniente a veces utilizar notación científica. La letra **E** (mayúscula o minúscula) indica que el número que está a su derecha es una potencia de 10.

Por ejemplo, 300 es lo mismo que 3×10^2. En notación científica escribiríamos 3E2. Análogamente, 0.03 es igual a 3×10^{-2}; en notación científica, 3E–2. Pruebe los siguientes ejemplos.

Puede escribir

```
?30*10 [INTRO]
 300
```

o bien

```
?3E1*1E1 [INTRO]
 300
```

?3000*1000 [INTRO] ... o bien ... **?3E3*1E3** [INTRO]

```
?3000*0.001 [INTRO] ... o bien ... ?3E3*1E-3 [INTRO]
 3
```

Parte 6: Grábelo en cinta ...

La memoria interna del CPC464 sólo puede «recordar» los datos mientras el ordenador está encendido; en terminología informática, es un medio de almacenamiento «volátil». Para que la información no se pierda cuando se apague la máquina, es necesario grabar los programas y los datos en un medio «no volátil», tal como la cinta o el disco (mediante la unidad de disco opcinal DDI-1).

Los programas se graban en la cinta con la orden **save**. Su forma más sencilla es la siguiente:

save "⟨*nombre del programa*⟩" [INTRO]

El ordenador responde con el mensaje

Press REC and PLAY then any key:

Este mensaje ('Pulse REC y PLAY y luego cualquier tecla') indica que se debe pulsar los botones **[REC]** y **[PLAY]** del magnetófono y después una tecla cualquiera en el teclado del ordenador. Por 'una tecla cualquiera' se debe entender cualquier tecla **menos las siguientes**: [MAYS], [FIJA MAYS], [CTRL] y [ESC].

Hecho esto, la cinta empieza a moverse. Al cabo de unos segundos aparece en la pantalla el siguiente mensaje:

Saving ⟨*nombre del programa*⟩ **block 1**

('Grabando ... bloque 1'). Cuando concluye la grabación, la cinta se detiene automáticamente y aparece en la pantalla el mensaje **Ready**. Pulse el botón **[STOP/EJECT]** del magnetófono para liberar la cinta.

Nota. El ordenador no tiene forma de saber si el usuario efectivamente ha pulsado los dos botones en el magnetófono. Por eso, si no los pulsa, o si sólo pulsa **[PLAY]**, el programa no quedará grabado.

Hay otras versiones de **save** que permiten grabar programas de BASIC en forma protegida, fichero de textos ASCII y ficheros binarios. Todas ellas están descritas en el capítulo 3.

Velocidades de grabación

El CPC464 puede grabar y leer las cintas a dos velocidades: en la velocidad *normal* la transferencia de datos se realiza a 1000 baudios; en la velocidad *alta*, a 2000 baudios.

Así pues, en velocidad alta la grabación y la lectura se realizan en la mitad de tiempo que en velocidad normal, pero se pierde el margen de seguridad que a veces es imprescindible cuando se trabaja con cintas de baja calidad o cuando la grabación y la lectura se realizan en ordenadores distintos y alguno de ellos tiene mal alineada la cabeza lectora/grabadora.

Si los programas se van a leer en la misma máquina que los graba, la alta velocidad es adecuada, siempre que las cintas no sean de calidad demasiado baja.

Si no se especifica otra cosa, el ordenador graba en velocidad normal. La orden

SPEED WRITE 1

selecciona velocidad alta. Para volver a velocidad normal se da la orden:

SPEED WRITE 0

Cuando carga un programa, el ordenador detecta a qué velocidad ha sido grabado y selecciona automáticamente la velocidad de lectura.

Parte 7: Introducción a los modos de pantalla, colores y gráficos

El AMSTRAD CPC464 puede funcionar en tres modos de pantalla: modo 0, modo 1 y modo 2.

Al encenderlo, el ordenador selecciona automáticamente el modo 1.

Para apreciar las diferencias entre los tres modos, encienda el ordenador y pulse la tecla del **1**. Manténgala pulsada hasta que se hayan llenado dos líneas de pantalla. Si cuenta los unos, observará que hay 40 en cada línea. Esto quiere decir que en modo 1 la pantalla tiene 40 columnas. Pulse [INTRO]: el ordenador responderá con un mensaje de **Syntax error**, pero no se preocupe; ésta es una forma rápida de obtener nuevamente el mensaje **Ready**, que indica que el ordenador está preparado para recibir nuestras instrucciones.

Escriba ahora:

mode 0 [INTRO]

Observará que ahora los caracteres son más grandes. Pulse otra vez la tecla del **1** y no la suelte hasta que se hayan llenado dos líneas de la pantalla. Compruebe que ahora hay 20 unos por línea. Esto significa que en modo 0 la pantalla tiene 20 columnas. Pulse nuevamente [INTRO] y luego escriba:

mode 2 [INTRO]

Observe que ahora los caracteres son muy pequeños. Puede comprobar que en modo 2 la pantalla tiene 80 columnas.

Resumiendo,

Modo 0 → 20 columnas
Modo 1 → 40 columnas
Modo 2 → 80 columnas

Finalmente, vuelva a pulsar [INTRO].

Colores

Este ordenador puede manejar 27 colores. En el monitor de fósforo verde (GT64) aparecen como distintas gradaciones de verde. Si usted adquirió el sistema con el monitor GT64, en cualquier momento puede comprar la unidad modulador/fuente de alimentación MP1, con la que podrá disfrutar de los colores del ordenador conectándolo a un televisor doméstico.

En modo 0 se puede visualizar simultáneamente 16 de los 27 colores disponibles.
En modo 1 se puede visualizar simultáneamente 4 de los 27 colores.
En modo 2 se puede visualizar simultáneamente 2 de los 27 colores.

Se puede controlar los colores del borde, del papel (fondo) y de la pluma (trazo) independientemente unos de otros.

Los 27 colores disponibles son los relacionados en la tabla 1, en la que incluimos también los números de referencia para la instrucción **INK** (tinta).

Para mayor comodidad, esta tabla aparece también en la carcasa del ordenador, al lado derecho.

Número de tinta	Color	Número de tinta	Color	Número de tinta	Color
0	Negro	9	Verde	18	Verde intenso
1	Azul	10	Cyan	19	Verde mar
2	Azul intenso	11	Azul celeste	20	Cyan intenso
3	Rojo	12	Amarillo	21	Verde lima
4	Magenta	13	Blanco	22	Verde pastel
5	Malva	14	Azul pastel	23	Cyan pastel
6	Rojo intenso	15	Anaranjado	24	Amarillo intenso
7	Morado	16	Rosado	25	Amarillo pastel
8	Magenta intenso	17	Magenta pastel	26	Blanco intenso

Tabla 1. Números de tinta y colores asociados.

Como hemos dicho, el ordenador se pone automáticamente en modo 1 al encenderlo. Para volver a modo 1 estando en cualquier otro, basta con escribir:

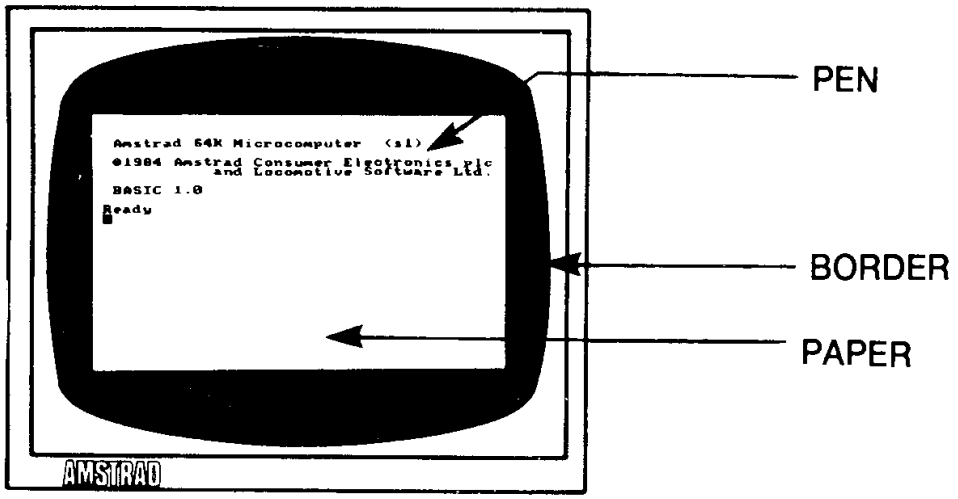
mode 1 INTRO

La pantalla

El borde (**BORDER**) es el área que rodea el papel (**PAPER**). (Cuando se enciende el ordenador, el borde y el papel son ambos azules.) Los caracteres que escribimos en la pantalla sólo puede estar en la región interior al borde. El *papel* es el fondo sobre el que la *pluma* (**PEN**) escribe los caracteres.

Vamos a explicar cómo se realiza la selección de los colores.

Cuando se enciende o reinicializa el ordenador, éste selecciona automáticamente el color número 1 para el borde. Como puede comprobar en la tabla 1, el color número 1 es el azul.



El color del borde se puede cambiar mediante la orden **BORDER** seguida del número del color deseado. Por ejemplo, escriba:

border 13 INTRO

Esto es fácil, pero en el interior de la pantalla las cosas son algo más complicadas:

Cuando se enciende o reinicializa la máquina, el número de papel seleccionado automáticamene es el 0, y el de la pluma es el 1. Esto no quiere decir que los colores correspondientes sean los que figuran en la tabla anterior con los números 0 y 1.

Lo que ocurre es que el 0 y el 1 son números de papel y de pluma, *no* números de tinta. Vamos a explicarnos. Supongamos que tenemos en nuestra mesa cuatro plumas, numeradas del 0 al 3, y 27 tinteros, numerados del 0 al 26. Es evidente que cuando decimos 'pluma número 1' no necesariamente estamos refiriéndonos al color número 1. La pluma número 1 puede estar cargada con tinta de cualquier color; de hecho, podemos tener las cuatro plumas cargadas con la misma tinta.

Pues bien, algo similar ocurre en el ordenador. Con la instrucción **PEN** (pluma) podemos elegir la pluma con la que vamos a escribir; la instrucción **INK** (tinta) nos permite 'cargar' esa pluma con tinta del color deseado.

Recordando que estamos en modo 1 (40 columnas), consulte la tabla 2 y comprobará que la pluma número 1 está cargada inicialmente con tinta de color 24. Pero el color 24 es, como podemos ver en la tabla 1, amarillo intenso: el color con que se escriben los caracteres cuando acabamos de encender el ordenador.

Las relaciones entre papel, pluma y tinta no son fijas. Los valores que se muestran en la tabla 2 son los seleccionados automáticamente en el momento de encender la máquina. Se los puede cambiar mediante la instrucción **INK** (tinta). Esta instrucción va seguida de dos números (parámetros). El primero es el número de papel o de pluma que vamos a cargar; el segundo es el color con que vamos a cargarlo. Los parámetros van separados por una coma.

Número de papel/pluma	Color de tinta Modo 0	Color de tinta Modo 1	Color de tinta Modo 2
0	1	1	1
1	24	24	24
2	20	20	1
3	6	6	24
4	26	1	1
5	0	24	24
6	2	20	1
7	8	6	24
8	10	1	1
9	12	24	24
10	14	20	1
11	16	6	24
12	18	1	1
13	22	24	24
14	Parpadeo 1,24	20	1
15	Parpadeo 16,11	6	24

Tabla 2. Asignación inicial de tintas a los papeles y las plumas.

Puesto que estamos utilizando la pluma número 1, vamos a cambiar el color de tinta correspondiente: la cargaremos con tinta de color anaranjado. Escriba:

ink 1,15 INTRO

Los caracteres han cambiado instantáneamente de color.

También vamos a cambiar el color del fondo mediante la instrucción INK. Sabemos que el número de papel seleccionado al encender la máquina es el 0; vamos a cambiar el color a verde (color número 9) escribiendo:

ink 0,9 INTRO

Ahora vamos a cambiar de pluma. Escriba:

pen 3 INTRO

El color de los caracteres ya escritos no ha cambiado. Esta instrucción afecta solamente a los que escribamos a continuación. En este momento estamos utilizando la pluma número 3. Como puede comprobar en las tablas 1 y 2, la tinta con que esta pluma está cargada inicialmente es la de color número 6 (rojo intenso). Para cambiarla a rosado escriba:

ink 3.16 INTRO

Recuerde que el 3 es el color de la pluma (seleccionado con la anterior instrucción **pen 3**) y que el 16 es el color de la tinta con que la cargamos (rosado).

Cambiemos ahora de papel. Cuando lo hagamos, el anterior color de fondo no cambiará, porque ese color ha sido impreso con otro papel. Escriba lo siguiente:

paper 2 [INTRO]

Consultando nuevamente las tablas 1 y 2, compruebe que el color inicialmente asignado al papel número 2 es cyan intenso. Cámbielo a negro escribiendo:

ink 2,0 [INTRO]

En este momento, en la pantalla tenemos caracteres escritos por las plumas 1 y 3, sobre fondos de papel números 0 y 2. También se puede cambiar el color de la tinta de una pluma o de un papel que no estén siendo utilizados. Por ejemplo, escriba la orden:

ink 1,2 [INTRO]

que cambia el color de todos los caracteres que habíamos escrito antes con la pluma número 1.

Escriba:

cls [INTRO]

para borrar la pantalla.

El lector ya debería ser capaz de hacer que el ordenador vuelva a los colores iniciales (borde y fondo azul y caracteres amarillos) utilizando las instrucciones **BORDER**, **PAPER**, **PEN** e **INK**. Inténtelo. Si no lo consigue, reinicialice el ordenador con [CTRL] [MAYS] [ESC].

Colores parpadeantes

Es posible hacer que una tinta cambie intermitentemente de color. Para ello se debe añadir otro número a la instrucción **INK** que asigna tintas a la pluma utilizada.

Vamos a hacer que los caracteres que escribamos alternen entre los colores blanco intenso y rojo intenso. Reinicialice la máquina con [CTRL] [MAYS] [ESC] y escriba lo siguiente:

ink 1,26,6 [INTRO]

En este caso el **1** es el número de la pluma, el **26** es el del color blanco intenso y el **6** es el del segundo color, rojo intenso.

El mismo efecto de parpadeo se puede dar a los colores del fondo, para lo cual se añade un segundo número de color a la instrucción **INK** que asigna tintas al papel actual. Para hacer que el color del fondo alterne entre verde y amarillo intenso escriba:

ink 0,9,24 [INTRO]

En este caso **0** es el número del papel, **9** es el número del color verde y **24** es el número del segundo color, el amarillo intenso.

Reinicialice el ordenador con [CTLR] [MAYS] [ESC].

Observe en la tabla 2 que en modo 0 hay dos números de pluma y dos números de papel, el **14** y el **15**, que tienen asignadas tintas parpadeantes. Es decir, sus correspondientes tintas están preprogramadas con un color adicional.

Escriba lo siguiente:

```
mode 0 [INTRO]
pen 15 [INTRO]
```

y verá en la pantalla la palabra **Ready** parpadeando entre azul celeste y rosado. Escriba ahora:

```
paper 14 [INTRO]
cls
```

El texto continúa parpadeando como antes, pero además el fondo ha empezado a alternar entre los colores amarillo y azul.

Los números de pluma y de papel **14** y **15** pueden ser reprogramados, mediante la adecuada instrucción **INK**, para que parpadeen con otras tintas o bien para asignarles un color fijo.

Finalmente, se puede hacer parpadear el borde sin más que especificar un segundo número de color en la instrucción **BORDER**. Escriba:

```
border 6,9 [INTRO]
```

El borde está parpadeando entre los colores rojo intenso y verde. Observe que al borde se le puede asignar uno o dos colores cualesquiera de los 27 disponibles, independientemente del modo en que esté la pantalla (0, 1 o 2).

Reinicialice el ordenador con [CTRL] [MAYS] [ESC].

Introduzca y ejecute el siguiente programa, que demuestra los colores disponibles.

```
10 MODE 0 [INTRO]
20 velocidad=600: REM establece velocidad del programa [INTRO]
30 FOR b=0 TO 26 [INTRO]
40 LOCATE 1,12 [INTRO]
50 BORDER b [INTRO]
60 PRINT "color del borde:";b [INTRO]
70 FOR t=1 TO velocidad [INTRO]
80 NEXT t,b [INTRO]
90 CLG [INTRO]
100 FOR p=0 TO 15 [INTRO]
110 PAPER p [INTRO]
```

```
120 PRINT "papel:";p [INTRO]
130 FOR n=0 TO 15 [INTRO]
140 PEN n [INTRO]
150 PRINT "pluma:";n [INTRO]
160 NEXT n [INTRO]
170 FOR t=1 TO velocidad*2 [INTRO]
180 NEXT t,p [INTRO]
190 MODE 1 [INTRO]
200 BORDER 1 [INTRO]
210 PAPER 0 [INTRO]
220 PEN 1 [INTRO]
230 INK 0,1 [INTRO]
240 INK 1,24 [INTRO]
run [INTRO]
```

OBSERVACIÓN MUY IMPORTANTE

En este programa, así como en otros capítulos y listados de este manual, las palabras clave de BASIC aparecen en mayúsculas, pues el ordenador convierte automáticamente minúsculas a mayúsculas cuando se le pide que haga un listado del programa (con **LIST**). En general, es preferible escribir las instrucciones en minúsculas; esto ayudará más tarde a detectar errores, ya que las palabras clave de BASIC que contengan algún error *no* serán convertidas a mayúsculas.

En lo que resta de este 'Curso de introducción' daremos los listados indistintamente en mayúsculas y minúsculas, para que usted se familiarice con esta característica de BASIC.

Los nombres de las variables, tales como **x** o **a$**, *no* son convertidos a mayúsculas al listar el programa. Sin embargo, el ordenador reconoce las variables tanto si están en mayúsculas como si están en minúsculas; de hecho, para el ordenador **x** es la misma variable que **X**.

Atención

A partir de ahora no vamos a seguir recordándole que debe pulsar [INTRO] al terminar de escribir cada línea de programa y cada orden directa, pues suponemos que usted ya está suficientemente habituado a hacerlo.

Gráficos

En la memoria del ordenador está pregrabado el diseño de cierto número de caracteres. Para escribirlos en la pantalla necesitamos la palabra clave:

chr$( )

Dentro de los paréntesis se pone el número del carácter, que debe estar en el margen de 32 a 255.

Reinicialice el ordenador con [CTRL] [MAYS] [ESC] y escriba lo siguiente:

print chr$(250)

No olvide pulsar [INTRO]. En la pantalla ha aparecido el carácter número **250**, que es la figura de un hombre caminando hacia la derecha.

Para ver todos los caracteres y símbolos, junto con su número correspondiente, introduzca y ejecute el siguiente programa (recuerde que debe pulsar [INTRO] al final de cada línea):

```
10 for n=32 to 255
20 print n;chr$(n);
30 next n
run
```

Todos estos caracteres y números figuran en el capítulo 'Para su referencia ...'.

LOCATE

Esta instrucción sirve para colocar el cursor en un lugar especificado de la pantalla. A menos que se lo mueva con una instrucción **LOCATE**, el punto de partida del cursor es el extremo superior izquierdo de la pantalla, punto de coordenadas **1,1** (en el sistema de coordenadas **x, y**, donde **x** es la posición horizontal e **y** es la vertical). En modo 1 hay 40 columnas y 25 filas (o líneas). Así, para colocar el cursor en el centro de la primera línea, tendremos que utilizar **20,1** como coordenadas **x, y**.

Teclee lo siguiente (sin olvidar pulsar [INTRO] al final de cada línea):

mode 1 ... borra la pantalla y lleva el cursor al extremo superior izquierdo

```
10 locate 20,1
20 print chr$(250)
run
```

Para comprobar que el carácter está en la primera línea, cambiemos el color del borde:

```
border 0
```

El borde es ahora negro y podemos ver que el hombrecillo está en el centro de la línea superior.

En modo 0 sólo hay 20 columnas, pero las mismas 25 líneas. Si ahora escribe:

```
mode 0
run
```

verá que el hombre se ha ido al extremo superior derecho de la pantalla. Esto ha ocurrido porque la coordenada **x** = 20 es la última columna en modo 0. Escriba:

```
mode 2
run
```

Vuelva a modo 1 con:

```
mode 1
```

Ahora pruebe usted con diferentes posiciones en **locate** y diferentes números en **chr$( )**. Por ejemplo, escriba:

```
locate 20,12:print chr$(240)
```

y verá una flecha en el centro de la pantalla. Observe que en esta instrucción:

20 es la coordenada x (horizontal) (margen: 1 a 40)
12 es la coordenada y (vertical) (margen: 1 a 25)
240 es el número del carácter (margen: 32 a 255)

Para mover el carácter 250 de izquierda a derecha de la pantalla, escriba lo siguiente:

```
10 CLS
20 FOR x=1 TO 39
30 LOCATE x,20
50 PRINT CHR$(250)
60 NEXT x
70 GOTO 10
run
```

Pulse [ESC] dos veces para abandonar el programa.

Si queremos borrar el carácter recién escrito antes de escribir el siguiente, debemos hacer

```
50 print " ";chr$(250)
```

(Esta nueva línea 50 reemplaza automáticamente la que teníamos antes.) Escriba:

```
run
```

Para mejorar la ilusión de movimiento del carácter por la pantalla, añada al programa anterior la siguiente línea:

```
40 call &bd19
```

Este programa se puede mejorar introduciendo pausas y utilizando un símbolo distinto para el retroceso.

Escriba:

```
list
```

y añada las siguientes líneas:

```
70 FOR n=1 TO 300:NEXT n
80 FOR x=39 TO 1 STEP -1
90 LOCATE x,20
100 FRAME
110 PRINT CHR$(251);" "
120 NEXT x
130 FOR n=1 TO 300:NEXT n
140 GOTO 20
run
```

PLOT

La instrucción **PLOT** es análoga a **LOCATE**, pero controla la posición del cursor gráfico y utiliza un sistema de coordenadas distinto, en el que las distancias se miden en *pixels* (un pixel, o *punto gráfico*, es la mínima área de la pantalla controlable individualmente).

El cursor gráfico es invisible, y distinto en todos los aspectos del cursor de texto.

La pantalla se divide en 640 puntos en horizontal por 400 en vertical. Las coordenadas **x, y** se toman con respecto al extremo inferior izquierdo de la pantalla, que es el punto de coordenadas **0, 0**. A diferencia de lo que ocurría con **LOCATE**, las coordenadas de este sistema no dependen del modo de pantalla (0, 1 o 2).

Para comprobarlo, reinicialice la máquina con [CTRL] [MAYS] [ESC] y escriba:

```
plot 320,200
```

Observe el punto que ha aparecido en el centro de la pantalla.

Hagamos lo mismo en modo 0:

```
mode 0
plot 320,200
```

El punto sigue estando en el centro de la pantalla, pero es más grande. Para ver el efecto en modo 2 escriba:

```
mode 2
plot 320,200
```

Como era de esperar, el punto sigue centrado, pero es mucho más pequeño.

Dibuje unos cuantos puntos en diversos lugares de la pantalla y en modos distintos para familiarizarse con esta instrucción. Cuando haya terminado, escriba:

```
mode 1
```

para volver a modo 1 y borrar la pantalla.

DRAW

Reinicialice la máquina con [CTRL] [MAYS] [ESC]. La instrucción **DRAW** dibuja una recta a partir de la posición actual del cursor gráfico. Para verla en acción, dibuje un rectángulo con el siguiente programa:

```
5 cls
10 plot 10,10
20 draw 10,390
30 draw 630,390
40 draw 630,10
50 draw 10,10
60 goto 60
run
```

En la línea 10 la instrucción **PLOT** coloca el cursor en el punto de partida deseado. A continuación, las instrucciones **DRAW** van dibujando segmentos de recta: hacia arriba, hacia la derecha, etc.

Pulse [ESC] dos veces para abandonar el programa.

(Observe la línea 60 de este programa. Esta línea establece un bucle infinito, del que sólo se sale deteniendo el programa. Una instrucción de este tipo es útil cuando se quiere evitar que el ordenador exhiba el mensaje **Ready** al terminar el programa.)

Añada al programa las siguientes líneas, las cuales dibujan un segundo rectángulo dentro del anterior:

```
60 plot 20,20
70 draw 20,380
80 draw 620,380
90 draw 620,20
100 draw 20,20
110 goto 110
run
```

Pulse [ESC] dos veces para detener el programa.

MOVE

Esta instrucción funciona igual que **PLOT** en cuanto a llevar el cursor gráfico a la posición especificada, pero *no* dibuja el punto.

Escriba:

```
cls
move 639,399
```

Aunque no podamos verlo, hemos llevado el cursor al extremo superior derecho de la pantalla.

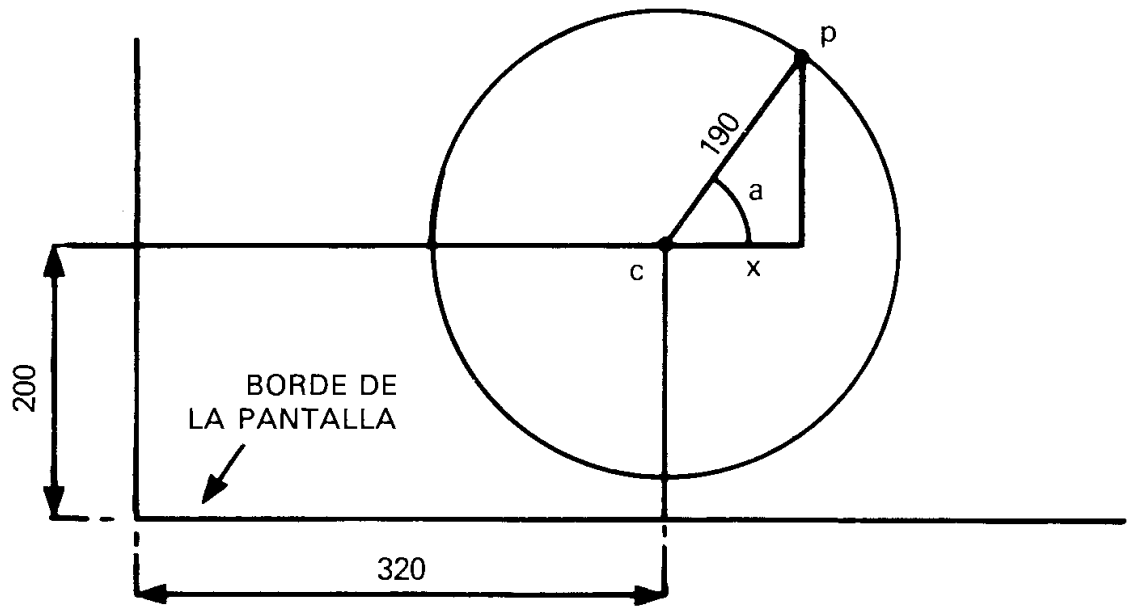
Para demostrarlo, vamos a dibujar una recta desde ese punto hasta el centro de la pantalla. Escriba:

```
draw 320,200
```

Circunferencias

Las circunferencias se pueden dibujar punto a punto o con trazo continuo. Una forma de dibujar una circunferencia consiste en dibujar puntos en las posiciones correctas. En la figura siguiente se ilustra cómo se calculan las coordenadas **x, y** de un punto **p** de la circunferencia. Las fórmulas son:

```
x=190*cos(a)
y=190*sin(a)
```



Dibujo de una circunferencia

En los programas anteriores hemos referido todos los dibujos al extremo inferior izquierdo de la pantalla. Si queremos que nuestra circunferencia quede centrada en la pantalla, debemos situar su centro en el punto de coordenadas **320,200**, y luego sumar a estos valores los dados por las fórmulas anteriores.

El siguiente programa dibuja punto a punto una circunferencia:

```
new
10 CLS
20 DEG
30 FOR a=1 TO 360
40 MOVE 320,200
50 PLOT 320+190*COS(a),200+190*SIN(a)
60 NEXT
run
```

Observe que hemos dado la orden **NEW** antes de introducir el programa. Esta orden hace que el ordenador borre el programa que en ese momento tenga en la memoria (de forma similar a lo que ocurre cuando se pulsa [CTRL] [MAYS] [ESC]). Sin embargo, la pantalla no se borra.

El radio de la circunferencia se puede reducir poniendo en lugar de 190 un número menor de pixels.

Para ver el efecto de dibujar la circunferencia de otra forma (en radianes), borre la línea 20 escribiendo:

```
20
```

En lugar de punto a punto, podemos dibujar la circunferencia con trazo continuo. Modifique la línea 50 para poner **draw** en lugar de **plot**. La línea será entonces:

```
50 draw 320+190*cos(a),200+190*sin(a)
```

Pruebe esta modificación con la línea 20 y sin ella.

Observe que en la línea 60 de este programa hemos puesto **NEXT** en lugar de **NEXT a**. En efecto, se puede omitir el nombre de la variable, pues el ordenador se encarga de averiguar a qué **FOR** corresponde cada **NEXT**. No obstante, en los programas en los que haya muchos bucles **FOR ... NEXT**, puede ser conveniente poner los nombres de las variables después de **NEXT** para que el programa sea más inteligible cuando se lo esté corrigiendo o estudiando.

ORIGIN

En el programa anterior utilizábamos la instrucción **MOVE** para establecer el centro de la circunferencia y luego sumábamos a las coordenadas del centro las coordenadas de los puntos de la circunferencia. Podemos evitar esas sumas si redefinimos la posición del origen de coordenadas con la instrucción **ORIGIN**. El ordenador entenderá las coordenadas que se le suministren a continuación como referidas al nuevo origen. Escriba lo siguiente:

```
new
10 cls
20 for a=1 to 360
30 origin 320,200
40 plot 190*cos(a),190*sin(a)
50 next
run
```

El siguiente programa dibuja cuatro circunferencias más pequeñas:

```
new
10 CLS
20 FOR a=1 TO 360
30 ORIGIN 196,282
40 PLOT 50*COS(a),50*SIN(a)
50 ORIGIN 442,282
60 PLOT 50*COS(a),50*SIN(a)
70 ORIGIN 196,116
80 PLOT 50*COS(a),50*SIN(a)
90 ORIGIN 442,116
100 PLOT 50*COS(a),50*SIN(a)
110 NEXT
run
```

Otra forma de dibujar la circunferencia (con trazo continuo) es la siguiente:

```
new
10 MODE 1
20 ORIGIN 320,200
30 DEG
40 MOVE 0,190
50 FOR a=0 TO 360 STEP 10
60 DRAW 190*SIN(a),190*COS(a)
70 NEXT
run
```

En este caso se van dibujando pequeños segmentos de recta, de un punto al siguiente de la circunferencia. Este método es mucho más rápido que el de dibujo punto a punto.

Observe una vez más el efecto de la instrucción **DEG**: suprima la línea 30 y ejecute el programa.

Un ejemplo . . .

Para concluir esta sección, le ofrecemos un programa de demostración de gráficos que incorpora muchas instrucciones que el lector ya debería entender.

```
10 MODE 1:BORDER 0:DEG
20 PRINT "Por favor, espere"
30 FOR n=1 to 3
40 INK 0,0:INK 1,26:INK 2,6:INK 3,18
50 IF n=1 THEN sa=120
60 IF n=2 THEN sa=135
70 IF n=3 THEN sa=150
80 IF n=1 THEN ORIGIN 0,-50,0,640,0,400 ELSE ORIGIN 0,0,0,640,0,400
90 DIM cx(5),cy(5),r(5),lc(5)
100 DIM np(5)
110 DIM px%(5,81),py%(5,81)
120 st=1:cx(1)=320:cy(1)=200:r(1)=80
130 FOR st=1 TO 4
140 r(st+1)=r(st)/2
150 NEXT st
160 FOR st=1 TO 5
170 lc(st)=0:np(st)=0
180 np(st)=np(st)+1
190 px%(st,np(st))=r(st)*SIN(lc(st))
200 py%(st,np(st))=r(st)*COS(lc(st))
```

```
210 lc(st)=lc(st)+360/r(st)
220 IF lc(st)<360 THEN 180
230 px%(st,np(st)+1)=px%(st,1)
240 py%(st,np(st)+1)=py%(st,1)
250 NEXT st
260 CLS:cj=REMAIN(1):cj=REMAIN(2)
270 cj=REMAIN(3):INK 1,2:st=1
280 GOSUB 350
290 LOCATE 1,1
300 EVERY 25,1 GOSUB 510
310 EVERY 15,2 GOSUB 550
320 EVERY 5,3 GOSUB 590
330 ERASE cx,cy,r,lc,np,px%,py%:NEXT
340 GOTO 340
350 cx%=cx(st):cy%=cy(st):lc(st)=0
360 FOR x%=1 TO np(st)
370 MOVE cx%,cy%
380 DRAW cx%+px%(st,x%),cy%+py%(st,x%),1+(st MOD 3)
390 DRAW cx%+px%(st,x%+1),cy%+py%(st,x%+1),1+(st MOD 3)
400 NEXT x%
410 IF st=5 THEN RETURN
420 lc(st)=0
430 cx(st+1)=cx(st)+1.5*r(st)*SIN(sa+lc(st))
440 cy(st+1)=cy(st)+1.5*r(st)*COS(sa+lc(st))
450 st=st+1
460 GOSUB 350
470 st=st-1
480 lc(st)=lc(st)+2*sa
490 IF (lc(st) MOD 360)<>0 THEN 430
500 RETURN
510 ik(1)=1+RND*25
520 IF ik(1)=ik(2) OR ik(1)=ik(3) THEN 510
430 INK 1,ik(1)
540 RETURN
550 ik(2)=1+RND*25
560 IF ik(2)=ik(1) OR ik(2)=ik(3) THEN 550
570 INK 2,ik(2)
580 RETURN
590 ik(3)=1+RND*25
600 IF ik(3)=ik(1) OR ik(3)=ik(2) THEN 590
610 INK 3,ik(3)
620 RETURN
```

Parte 8: Sonidos

El ordenador emite su sonido a través del altavoz que tiene incorporado. Si está utilizando la unidad modulador/fuente de alimentación MP1 y un televisor doméstico, baje el mando de volumen del televisor al mínimo.

El nivel de sonido se ajusta con el mando **VOLUMEN** que hay en el panel derecho del ordenador. También es posible enviar el sonido a un amplificador externo, por medio de la salida marcada con **E/S** en el ordenador. De esta forma se puede oír el sonido en estereofonía, utilizando los altavoces o los auriculares de un equipo de alta fidelidad. En la parte 2 de este 'Curso de introducción' se explica cómo tomar las señales de la salida **E/S** del ordenador.

La instrucción SOUND

La instrucción **SOUND** tiene siete parámetros. Los dos primeros son imprescindibles, mientras que los restantes son opcionales. La forma de la instrucción es la siguiente:

SOUND ⟨*situación de canales*⟩,⟨*periodo de tono*⟩,⟨*duración*⟩,⟨*volumen*⟩,
⟨*envolvente de volumen*⟩,⟨*envolvente de tono*⟩,⟨*periodo de ruido*⟩

Parece complicada a primera vista, pero veremos que en realidad no lo es tanto cuando hayamos analizado el significado de todos los parámetros. Así pues, estudiemos los parámetros uno por uno.

Situación de los canales

Para no complicar las cosas, por ahora nos limitaremos a considerar este parámetro como simple selector de canales. Hay tres canales de sonido; de momento sólo utilizaremos el número **1**.

Periodo de tono

El periodo de tono es la forma técnica de especificar el tono del sonido; en otras palabras, de identificar la nota (DO, RE, MI, FA, etc.). Cada nota tiene un número de identificación, que es el periodo de tono. Si consulta el capítulo 'Para su referencia . . .', comprobará que la nota DO media tiene un periodo de tono de **478**.

Reinicialice el ordenador con [CTRL] [MAYS] [ESC] y luego escriba:

```
10 sound 1,478
run
```

Acaba de oír la nota DO media, que ha sonado durante 0.2 segundos.

¿No ha oído nada? Suba el control de volumen del ordenador y vuelva a ejecutar el programa.

Duración

Este parámetro especifica la duración del sonido. Las unidades son de centésimas de segundo (0.01 s). Si no se especifica duración, el ordenador toma el valor implícito para este parámetro, que es **20**; por eso la nota que acaba de oír ha durado $0.01 \times 20 = 0.2$ segundos.

Si queremos que la nota dure 1 segundo, el valor del parámetro tendrá que ser **100**; para dos segundos, **200**; etc. Escriba lo siguiente:

```
10 sound 1,478,200
run
```

Acaba de oír la nota DO media con duración de 2 segundos.

Volumen

Este parámetro especifica el volumen de la nota. El margen de valores va de **0** a **15**. El 'volumen' **0** es el mínimo; el **15** es el máximo. Si no se especifica ningún valor, el ordenador toma el implícito, que es **12**. Escriba lo siguiente:

```
10 sound 1,478,200,5
run
```

Fíjese en el volumen de este sonido y compárelo con el siguiente:

```
10 sound 1,478,200,15
run
```

Como puede observar, es mucho más intenso que el anterior.

Envolvente de volumen

Se puede hacer que el volumen de una nota no sea constante, sino que varíe con el tiempo mientras la nota está sonando. Para especificar la forma de variación del volumen en función del tiempo necesitamos otra instrucción: **ENV**. De hecho, podemos definir varias en-

volventes de volumen distintas, cada una de las cuales tendrá su número de referencia. Así, si hemos definido la envolvente de volumen número **1**, podemos utilizarla poniendo en la instrucción **SOUND** el número **1** como parámetro de envolvente de volumen. En seguida explicaremos cómo se definen las envolventes de volumen.

Envolvente de tono

Se puede hacer que el tono de una nota no sea constante, sino que varíe con el tiempo mientras la nota está sonando. Para especificar la forma de variación del tono en función del tiempo necesitamos otra instrucción: **ENT**. De hecho, podemos definir varias envolventes de tono distintas, cada una de las cuales tendrá su número de referencia. Así, si hemos definido la envolvente de tono número **1**, podemos utilizarla poniendo en la instrucción **SOUND** el número **1** como parámetro de envolvente de tono. En seguida explicaremos cómo se definen las envolventes de tono.

Ruido

El periodo de ruido es el último parámetro de la instrucción **SOUND**. Las características del ruido se pueden variar eligiendo para este parámetro un valor comprendido entre **1** y **31**. Ponga el valor **2** en la instrucción **SOUND** y escuche el efecto. Ponga luego el número **27** y observe la diferencia. Escriba:

```
10 sound 1,478,200,15,,,2
```

En esta instrucción hemos dejado en blanco dos parámetros (,,,) porque aún no hemos creado ninguna envolvente de volumen ni de tono.

Definición de una envolvente de volumen

La instrucción que define envolventes de volumen es **ENV**. En su versión más sencilla, esta instrucción lleva 4 parámetros. Su forma es la siguiente:

ENV ⟨*número de envolvente*⟩,⟨*número de escalones*⟩,⟨*altura de cada escalón*⟩, ⟨*duración de cada escalón*⟩

Estudiemos los parámetros uno por uno.

Número de la envolvente

Es el número de referencia (entre **0** y **15**) por el que la citaremos en la instrucción **SOUND**.

Número de escalones

Este parámetro especifica en cuántas etapas queremos que la nota evolucione antes de terminar. Por ejemplo, si la nota ha de durar 10 segundos, podemos dividirla en 10 etapas de 1 segundo cada una; en tal caso, el parámetro 'número de escalones' tendría el valor **10**.

El margen de este parámetro es de **0** a **127**.

Altura de cada escalón

En cada etapa, el volumen puede variar con respecto al nivel anterior en un número de unidades comprendido en **0** y **15**. Estos 16 niveles de sonido son los mismos que se definen en la instrucción **SOUND**. Sin embargo, el margen del parámetro 'altura de escalón' es de **−128** a **+127**; el nivel del volumen vuelve a **0** cada vez que sobrepasa el **15**.

Duración de cada escalón

Este parámetro especifica la duración de cada escalón en unidades de centésimas de segundo. El margen de valores es de **0** a **255**; esto significa que la duración máxima de cada escalón es de 2.56 segundos (el **0** se considera como **256**).

Por consiguiente, el producto del parámetro 'número de escalones' por el parámetro 'duración de cada escalón' no debería ser mayor que el parámetro 'duración' especificado en la instrucción **SOUND**. De lo contrario, el sonido terminará antes de que se hayan completado todos los escalones de la envolvente. (En ese caso el ordenador ignora el resto del contenido de la envolvente.)

Análogamente, si la duración especificada en **SOUND** es mayor que la impuesta por el producto de 'número de escalones' por 'duración de cada escalón', la nota continuará sonando aunque se hayan terminado los escalones de la envolvente de volumen, y lo hará al nivel del último escalón de la envolvente.

Para practicar con las envolventes de volumen, pruebe el siguiente programa:

```
10 env 1,10,1,100
20 sound 1,284,1000,1,1
run
```

La línea 20 especifica un sonido con número de tono igual a **284** (LA internacional), duración de 10 segundos, volumen inicial igual a **1** y envolvente de tono número **1**, la cual consiste en **10** escalones de altura **1** y duración 1 segundo (**100**×0.01).

Pruebe las siguientes formas de la línea 10 y trate de escuchar el efecto de las diferentes envolventes de volumen:

```
10 env 1,100,1,10
10 env 1,100,2,10
```

```
10 env 1,100,4,10
10 env 1,50,20,20
10 env 1,50,2,20
10 env 1,50,15,30
```

Finalmente, pruebe la siguiente:

```
10 env 1,50,2,10
```

Observe que el volumen permanece constante durante la segunda mitad de la nota. Esto ocurre por que el número de escalones es **50** y la duración de cada uno de ellos es 0.1 s, con lo que la duración total de la envolvente es de solamente 5 segundos, mientras que en la instrucción **SOUND** se especifica una duración de 10 segundos.

Haga usted otras pruebas y trate de generar sonidos diferentes.

Si quiere crear envolventes más complejas, puede repetir los tres últimos parámetros de la instrucción **ENV** para definir otras 'secciones' de la envolvente (hasta un máximo de 4 secciones).

Definición de una envolvente de tono

La instrucción que define envolventes de tono es **ENT**. En su versión más sencilla, esta instrucción lleva 4 parámetros. Su forma es la siguiente:

ENT ⟨*número de envolvente*⟩,⟨*número de escalones*⟩,⟨*altura de cada escalón*⟩, ⟨*duración de cada escalón*⟩

Como siempre, estudiemos los parámetros uno por uno.

Número de la envolvente

Es el número de referencia (entre **0** y **15**) por el que la citaremos en la instrucción **SOUND**.

Número de escalones

Este parámetro especifica en cuántas etapas queremos que la nota evolucione antes de terminar. Por ejemplo, si la nota ha de durar 10 segundos, podemos dividirla en **10** etapas de 1 segundo cada una; en tal caso, el parámetro 'número de escalones' tendría el valor **10**.

El margen de este parámetro es de **0** a **239**.

Periodo de tono de cada escalón

En cada etapa, el tono puede variar con respecto al de la anterior en un número de unidades comprendido entre **−128** y **+127**. Las variaciones negativas representan aumento del tono (tono más agudo); las variaciones positivas reducen el tono (tono más grave). El valor mínimo del período de tono es **0**. Téngalo en cuenta cuando programe envolventes de tono. En el capítulo 'Para su referencia . . .' se da la lista completa de los periodos de tono.

Duración de cada escalón

Este parámetro especifica la duración de cada escalón en unidades de centésimas de segundo. El margen de valores es de **0** a **255**; esto significa que la duración máxima de cada escalón es de 2.56 segundos (el **0** se considera como **256**).

Por consiguiente, el producto del parámetro 'número de escalones' por el parámetro 'duración de cada escalón' no debería ser mayor que el parámetro 'duración' especificado en la instrucción **SOUND**. De lo contrario, el sonido terminará antes de que se hayan completado todos los escalones de la envolvente. (En ese caso el ordenador ignora el resto del contenido de la envolvente.)

Análogamente, si la duración especificada en **SOUND** es mayor que la impuesta por el producto de 'número de escalones' por 'duración de cada escalón', la nota continuará sonando aunque se hayan terminado los escalones de la envolvente de tono, y lo hará con el tono correspondiente al de la última etapa de la envolvente.

Para practicar con las envolventes de tono, pruebe el siguiente programa:

```
10 ent 1,100,2,2
20 sound 1,284,200,15,,1
run
```

La línea 20 especifica un sonido con número de tono igual a **284** (LA internacional), duración de 2 segundos, volumen inicial igual a **15** (el máximo), sin envolvente de volumen (parámetro en blanco ,,) y con envolvente de tono número **1**.

En la línea 10 se define la envolvente de tono, la cual consiste en **100** escalones de altura **2** (reducción del tono) y duración 0.02 segundos (**2**×0.01).

Pruebe las siguientes formas de la línea 10 y trate de escuchar el efecto de las diferentes envolventes de tono:

```
10 ent 1,100,−2,2
10 ent 1,10,4,20
10 ent 1,10,−4,20
```

Cambie ahora la instrucción **sound** y la envolvente de tono escribiendo:

```
10 ent 1,2,17,70
20 sound 1,142,140,15,,1
30 goto 10
run
```

Pulse [ESC] dos veces para detener el programa.

Se puede combinar las instrucciones **ENV**, **ENT** y **SOUND** para crear sonidos más complejos. Empiece con el siguiente programa:

```
10 env 1,100,1,3
20 ent 1,100,5,3
30 sound 1,284,300,1,1,1
run
```

Cambie la línea 20 por la siguiente:

```
20 ent 1,100,-2,3
run
```

Y finalmente pruebe el siguiente programa:

```
10 env 1,100,2,2
20 ent 1,100,-2,2
30 sound 1,284,200,1,1,1
run
```

Si quiere crear envolventes más complejas, puede repetir los tres últimos parámetros de la instrucción **ENT** para definir otras 'secciones' de la envolvente (hasta un máximo de 4 secciones).

Haga usted mismo otras pruebas. Por ejemplo, incluya ruido en la instrucción **SOUND** y añada más secciones a las envolventes de volumen y de tono.

En el capítulo 'Lista completa de las palabras clave del BASIC de AMSTRAD' explicaremos con todo detalle las diversas instrucciones de sonido. Si le interesan los aspectos más melódicos del sonido, consulte la sección 'El sonido de la música' del capítulo titulado 'Cuando usted guste . . .'.

Capítulo 2
La génesis de un programa

Pues bien. Usted ya ha leído el 'Curso de introducción' y tiene ante sí el ordenador encendido. Ya sabe cómo hacer que el ordenador repita una tarea varias veces con la instrucción **FOR ... NEXT** y cómo hacerle realizar operaciones de forma condicional con la instrucción **IF ... THEN ...** .

Pero pronto empezará a cansarse de ver su nombre repetido hasta la saciedad en la pantalla y querrá aprender algo de informática más seria, algo útil o divertido. En el capítulo siguiente damos la lista de todas las palabras clave del BASIC de AMSTRAD, junto con su sintaxis, es decir, la forma de utilizarlas. Con estas herramientas en la mano, el único límite a lo que puede obtener del ordenador lo fijará su imaginación.

Si no ha utilizado antes ningún otro ordenador, es probable que la palabra 'programación' le suene amenazadora. ¡No le tenga miedo! Programar es más fácil de lo que usted cree, y desde luego mucho más fácil de lo que la jerga técnica da a entender.

No se deje asustar por BASIC y verá qué pronto empieza a disfrutar de la programación y de los frutos de su esfuerzo. Programar puede llegar a ser un ejercicio muy gratificante, en particular para el principiante que se pone a experimentar con la máquina y el lenguaje. Recuerde siempre que, a excepción del riesgo de borrar accidentalmente una cinta, nada que usted haga *en el teclado* puede dañar el ordenador; siempre vale la pena intentar algo nuevo.

¿Por dónde empezamos?

En programación, como en muchas otras actividades, lo más difícil es empezar. Sin embargo, lo peor que puede hacer es lanzarse directamente al teclado sin antes pensar para qué.

El primer paso en el desarrollo de un programa debe ser establecer claramente qué se pretende que el programa haga y cómo se quiere que presente los resultados en la pantalla.

Decidido esto, se puede empezar a escribir un programa que realice las funciones deseadas, preocupándose siempre de que el programa discurra suavemente de principio a fin, con el mínimo número de saltos posible. Un buen programa debe ser fácil de leer una vez listado, no un lío indescifrable que haga imposible la detección de errores o la depuración del programa.

Afortunadamente, BASIC es un lenguaje muy tolerante que facilita la depuración de los programas emitiendo mensajes que indican dónde está el error y en qué consiste. Además, hace posible la ampliación de los programas, pues permite intercalar líneas nuevas entre las ya existentes.

Un programa sencillo

Manos a la obra. Vamos a escribir un programa que controle y mantenga un fichero con los nombres y números de teléfono de nuestros amigos. Le llamaremos 'guía de teléfonos'. Para respetar las reglas que hemos enunciado, '¿qué debe hacer el programa?'; '¿cómo debe presentar los resultados en la pantalla?'

El programa debe permitirnos almacenar 100 nombres y números de teléfono. Cuando queramos averiguar un número de teléfono, escribiremos el nombre y el ordenador nos dará el número. Además, por si no recordamos cómo habíamos escrito algún nombre, el programa debe ser capaz de mostrar toda la información en la pantalla. Dicho sea de paso, esto ya nos hace empezar a pensar en la forma de presentación de los resultados en la pantalla.

De acuerdo, vamos al teclado. Empezaremos por escribir el nombre del programa:

```
10 REM guia de telefonos
```

No es imprescidible dar nombre a todos los programas, pero comprobará cuando tenga grabados unos cuantos que es muy útil poder averiguar de un golpe de vista cuál es cuál.

Lo siguiente que necesitamos es poder captar por el teclado (con **INPUT**) una cadena de caracteres (quizá un nombre) para luego poder asignarlo a una variable. A la variable que represente el nombre le llamaremos **NOMBRE$**. Lo mismo se puede hacer con el número de teléfono; a la variable le llamaremos **TEL$**.

¿Recuerda todavía los programas del 'Curso de Introducción'? En algunos de ellos utilizábamos la instrucción **INPUT** para captar los valores de las variables. Escribamos:

```
20 INPUT "Escriba el nombre:";NOMBRE$
30 INPUT "Escriba el numero de telefono:";TEL$
run
```

En respuesta al programa podemos introducir un nombre (por ejemplo, **Juan**) y luego un número de teléfono (por ejemplo, **923 233717**).

El programa ha almacenado la información que le hemos sumistrado, pero no ha escrito ningún resultado en la pantalla. Así pues, en algún lugar del programa necesitaremos instrucciones del tipo

PRINT NOMBRE$** ... y ... **PRINT TEL$

con las que recuperemos los valores de las variables.

Algo falla, ¿verdad? Dijimos que nuestro programa debería ser capaz de almacenar 100 nombres con sus números de teléfono; si seguimos por este camino, necesitaremos 200 instrucciones **INPUT** para escribrir la lista en la pantalla. No se inquiete; los ordenadores tienen un recurso que permite manejar con facilidad grandes cantidades de datos: se trata de las 'listas'. Una lista es una variable que puede representar, con un solo nombre (tal co-

mo **NOMBRE$**), un grupo de datos (en nuestro caso necesitamos 100) que se identifican por un *índice*. Cuando queramos referirnos a un dato concreto, lo citaremos por el nombre de la variable seguido del índice entre paréntesis; por ejemplo, **NOMBRE$(27)**, donde **27** es el índice y **NOMBRES$** es lo que denominamos *variable indexada*. Si como índice ponemos una variable numérica, **NOMBRE$(x)**, en lugar de un número concreto, podemos manejar la lista completa de datos, del 1 al 100, variando el valor de **x** dentro de un bucle **FOR ... NEXT** (v. g., **FOR x=1 TO 100**). A medida que se va incrementando el valor de **x**, el índice de **NOMBRE$** va señalando los diferentes elementos de la lista.

En nuestro programa necesitamos dos listas, **NOMBRE$** y **TEL$**, cada una con capacidad para 100 elementos. Pero tenemos que informar al ordenador de nuestras intenciones, y para ello debemos *dimensionar* las listas con la instrucción **DIM**. Cambiemos las líneas 20 y 30:

```
20 DIM NOMBRE$(100)
30 DIM TEL$(100)
```

Ahora ya podemos escribir el tramo del programa que nos va a permitir introducir nombres y números de teléfono. Añada las siguientes líneas:

```
40 FOR x=1 TO 100
50 INPUT;" Nombre ";NOMBRE$(x)
60 INPUT " Telefono ";TEL$(x)
70 NEXT
run
```

Pero no es probable que vayamos a introducir los 100 nombres de una vez, y además la forma de presentación en la pantalla es muy desordenada. Tenemos que organizarnos mejor. En primer lugar, antes de captar un nombre, vamos a librarnos de todo el texto superfluo que tenemos en la pantalla:

```
45 cls
```

¿Qué podemos hacer para decirle al ordenador que de momento ya no queremos introducir más datos? Podemos detener el programa pulsando [ESC] dos veces, pero cuando volvamos a teclear **run** perderemos todos los datos introducidos anteriormente.

Ésta es la solución. Cada vez que el programa capte un nuevo nombre, haremos que examine el dato introducido para ver si consta de al menos un carácter o si, por el contrario, es la 'cadena vacía'. Si lo es, haremos que suspenda la captación de datos. Añada lo siguiente:

```
55 IF NOMBRE$(x)="" THEN 80
80 PRINT "Fin de la entrada de datos"
```

También necesitamos que el ordenador indique al usuario qué debe hacer para suspender la introducción de datos. Añada, pues, la siguiente línea:

```
47 PRINT "Pulse [INTRO] para terminar"
```

Ahora vamos a escribir las instrucciones que exhiban en la pantalla la información almacenada. Primero la lista completa:

```
90 FOR x=1 TO 100
100 PRINT NOMBRE$(x);" ";TEL$(x)
110 NEXT
```

Otra vez nos ocurre que el programa no sabe dónde pararse y continúa hasta el centésimo elemento. Esto tiene fácil arreglo:

```
95 IF NOMBRE$(x)="" THEN 120
120 PRINT "Fin de la lista"
```

La línea 95 comprueba si **NOMBRE$(x)** es la cadena vacía; si lo es, concluye la escritura de la lista saltando a la línea 120.

Atendamos a los otros requisitos que habíamos formulado al idear el programa. Necesitamos unas instrucciones que busquen en la lista un nombre introducido por el ususario. Escriba lo siguiente:

```
130 INPUT "Buscar ";BUSCA$
140 FOR x=1 TO 100
150 IF INSTR(NOMBRE$(x),BUSCA$)=0 THEN 180
160 PRINT NOMBRE$(x);" ";TEL$(x)
170 END
180 NEXT
190 PRINT"No lo encuentro"
```

En la línea 150 hay una instrucción nueva: **INSTR**. Esta instrucción determina si la cadena literal **BUSCA$** (que es el nombre que estamos buscando, captado en la línea 130) está contenida en **NOMBRE$(x)**. Si no lo está, **INSTR** genera el valor **0**, y entonces el programa salta a la línea 180 para probar con el siguiente valor de **x**. Si el programa ha examinado sin éxito todos los elementos de la lista, continúa en la línea 190, donde informa de lo que ha ocurrido. En cambio, si encuentra el nombre, **INSTR** produce un valor mayor que **0**, lo que permite que el programa pase de la línea 150 a la 160 para escribir el nombre y el número de teléfono y luego terminar con la instrucción **END** de la línea 170.

Como ve, nuestro programa avanza muy deprisa, pero todavía nos queda mucho por hacer. Meditemos un poco sobre los inconvenientes de este programa, empezando por el orden en que funciona: primero capta la información, después la escribe completa en la pantalla, luego busca un nombre concreto.

¿Qué ocurre si . . . ?

Precisamente. ¿Qué ocurre si no queremos que los acontecimientos se produzcan en ese orden? Podríamos querer consultar un teléfono que introdujimos ayer. O podríamos querer

ampliar la lista con otros nombres. Esto nos plantea nuevos problemas, para los que tenemos que encontrar solución. Después de todo, en esto es en lo que consiste la programación. Ya hemos dicho que BASIC es muy amable y nos permite rectificar e incluir nuevas instrucciones con gran facilidad; pero un buen programador habría pensado en todo esto de antemano.

Otro grave inconveniente del programa consiste en que las listas quedan almacenadas solamente en la memoria, la cual se borra cada vez que se apaga el ordenador. De poco nos serviría este programa si tuviéramos que escribir todos los datos cada vez que lo ejecutásemos; así pues, necesitamos que el programa pueda grabar los valores de los elementos de **NOMBRE$** y **TEL$** antes de que apaguemos el ordenador, y también que pueda cargar los datos cada vez que lo ejecutemos.

Soluciones

El primero de los problemas mencionados (el del orden de ejecución) se resuelve haciendo que el programa dé la posibilidad de elegir entre las funciones que sabe realizar. De un programa de este tipo se dice que está 'controlado por menú' pues, en efecto, ofrece un menú de opciones de entre las cuales se elige la deseada. Si alguna vez ha sacado dinero de un cajero automático, ya ha utilizado un programa de ordenador controlado por menú. Añadamos un menú a nuestro programa:

```
32 PRINT"1. Introd. datos"
33 PRINT"2. Listado"
34 PRINT"3. Busqueda"
35 PRINT"4. Grabar de datos"
36 PRINT"5. Cargar datos"
37 INPUT "Elija una opción (1-5) ";el
38 ON el GOSUB 40,90,130

85 RETURN
125 RETURN
170 RETURN
200 RETURN
```

Hemos hecho que el programa escriba la lista de opciones y que capte la elección del usuario mediante una instrucción **INPUT** para asignarla a la variable **el**. La instrucción **ON el GOSUB** de la línea 38 funciona de la siguiente forma: si **el=1**, invoca la subrutina de la línea 40; si **el=2**, invoca la subrutina de la línea 90, etc.

Puesto que todas las funciones están ahora en subrutinas (controladas por la instrucción **ON el GOSUB**), tienen que terminar en una instrucción **RETURN**. De ahí que hayamos incluido las líneas 85, 125, 170 y 200.

¿Recuerda para qué sirve **RETURN**? Hace que BASIC retorne de la subrutina a la instrucción siguiente a aquella que invocó la subrutina. En nuestro caso, el punto de retorno es

la instrucción siguiente a la línea 38, que es la línea 40. Si dejamos que esto ocurra, el programa entra en la subrutina de 'Introducción de datos'. Para impedirlo añadimos la línea

```
39 GOTO 32
```

que reconduce el programa al menú.

Ejecute el programa y vea cuánto hemos progresado.

Bien; vamos a echar un vistazo al listado. (Detenga el programa pulsando [ESC] dos veces.) Escriba:

```
LIST
```

Esto es lo que tenemos por el momento:

```
10 REM guia de telefonos
20 DIM NOMBRE$(100)
30 DIM TEL$(100)
32 PRINT"1. Introd. datos"
33 PRINT"2. Listado"
34 PRINT"3. Busqueda"
35 PRINT"4. Grabar datos"
36 PRINT"5. Cargar datos"
37 INPUT "Elija una opcion (1-5) ";el
38 ON el GOSUB 40,90,130
39 GOTO 32
40 FOR x=1 TO 100
45 CLS
47 PRINT"Pulse [INTRO] para terminar"
50 INPUT;" Nombre ";NOMBRE$(x)
55 IF NOMBRE$(x)="" THEN 80
60 INPUT;" Telefono ";TEL$(x)
70 NEXT
80 PRINT "Fin de la entrada de datos"
85 RETURN
90 FOR x=1 TO 100
95 IF NOMBRE$(x)="" THEN 120
100 PRINT NOMBRE$(x);" ";TEL$(x)
110 NEXT
120 PRINT "Fin de la lista"
125 RETURN
130 INPUT "Buscar ";BUSCA$
140 FOR x=1 TO 100
150 IF INSTR(NOMBRE$(x),BUSCA$)=0 THEN 180
160 PRINT NOMBRE$(x);" ";TEL$(x)
170 END
```

```
180 NEXT
190 PRINT"No lo encuentro"
200 RETURN
```

Observe que en algunas zonas del programa las instrucciones están muy apretadas y no dejan hueco para introducir otras nuevas. Reorganicemos el programa renumerando las líneas. Escriba:

```
RENUM
LIST
```

El listado es ahora:

```
10 REM guia de telefonos
20 DIM NOMBRE$(100)
30 DIM TEL$(100)
40 PRINT"1. Introd. datos"
50 PRINT"2. Listado"
60 PRINT"3. Busqueda"
70 PRINT"4. Grabar datos"
80 PRINT"5. Cargar datos"
90 INPUT "Elija una opcion (1-5) ";el
100 ON el GOSUB 120,210,270
110 GOTO 40
120 FOR x=1 TO 100
130 CLS
140 PRINT"Pulse [INTRO] para terminar"
150 INPUT;" Nombre ";NOMBRE$(x)
160 IF NOMBRE$(x)="" THEN 190
170 INPUT;" Telefono ";TEL$(x)
180 NEXT
190 PRINT "Fin de la entrada de datos"
200 RETURN
210 FOR x=1 TO 100
220 IF NOMBRE$(x)="" THEN 250
230 PRINT NOMBRE$(x);" ";TEL$(x)
240 NEXT
250 PRINT "Fin de la lista"
260 RETURN
270 INPUT "Buscar ";BUSCA$
280 FOR x=1 TO 100
290 IF INSTR(NOMBRE$(x),BUSCA$)=0 THEN 320
300 PRINT NOMBRE$(x);" ";TEL$(x)
310 END
320 NEXT
330 PRINT"No lo encuentro"
340 RETURN
```

¡Mucho más limpio! Pero sigamos con el programa. Ahora queremos que cada vez que introduzcamos un dato nuevo (nombre y número), el ordenador lo coloque al final de la lista actual, en el primer lugar que encuentre vacío. En esta ocasión vamos a utilizar la instrucción **LEN** para determinar la longitud de una cadena literal. Especificaremos lo siguiente:

Si **(IF)** la longitud de **NOMBRE$(x)** es mayor que **0**, es decir, si ya hay un dato en ese elemento de la lista, entonces **(THEN)** ir a la línea 180 (que provoca el salto al siguiente elemento de la lista):

```
135 IF LEN(NOMBRE$(x))>0 THEN 180
```

Una solución sencilla, ¿verdad? Los problemas de este tipo siempre tienen solución si conocemos bien el repertorio de instrucciones y nos paramos a pensar un poco. Para casi todas las necesidades hay casi siempre al menos una instrucción; cuanto mayor es la experiencia en programación, más inmediatas son las soluciones.

Vamos con otro requisito del programa: grabar los valores de las variables para poder cargarlos más tarde. En la parte 6 del 'Curso de introducción' hemos explicado cómo grabar programas (con la orden **SAVE**). Pero el programa no es más que la herramienta que nos permite captar y manipular los datos; cuando grabamos el programa sólo estamos guardando la herramienta, no los valores de las variables.

Así pues, tenemos que escribir una sección del programa que grabe los valores de las variables. Para ello crearemos un 'fichero de datos'.

Lo primero es abrir (**OPEN**) un fichero en dirección de salida (**OUT**) dándole el nombre de '**datos**'. Después escribiremos (**WRITE**) los valores de las variables **NOMBRE$(x)** y **TEL$(x)**, de **1** a **100**, en el fichero; finalmente, cerraremos (**CLOSE**) el fichero y retornaremos (**RETURN**) al menú. Pondremos esta rutina en las líneas 350 y siguientes. Para no tener que escribir los números de línea, ejecutaremos la orden

```
AUTO 350
```

que inicia la numeración automática de las líneas a partir del número 350:

```
350 OPENOUT "!datos"
360 FOR x=1 TO 100
370 WRITE #9,NOMBRE$(x),TEL$(x)
380 NEXT
390 CLOSEOUT
400 PRINT"Fin de la grabación de los datos"
410 RETURN
```

Cuando haya terminado de teclear la línea 410 y haya pulsado [INTRO], pulse [ESC] para interrumpir la función **AUTO**.

Como tenemos una rutina más, tendremos que poner otro número de línea en la instrucción **ON el GOSUB** de la línea 100. Para ello vamos a corregir (editar, **EDIT**) la línea 100:

```
100 ON el GOSUB 120,210,270,350
```

A partir de ahora, cada vez que elijamos la opción 4 del menú, el programa grabará el fichero de datos en la cinta.

Observe la línea 370, en la que se escriben los valores de **NOMBRE$(x)** y **TEL$(x)** en la cinta: la expresión **#9** que aparece después de **WRITE** es un 'indicador de canal' que especifica a qué canal debe el ordenador enviar los datos. El ordenador maneja 10 canales:

Los ocho primeros (**#0** a **#7**) dirigen los datos a la pantalla [bien a la pantalla completa, bien a alguna ventana (**WINDOW**)].

El canal **#8** es el de la impresora.

Finalmente, el canal **#9** es el que conecta el ordenador con el magnetófono; éste es el que utilizamos en la instrucción 370.

Ya hemos escrito las instrucciones que graban los datos en la cinta. Ahora nos falta ocuparnos de la lectura de los datos para ponerlos a disposición del programa. Como vamos a escribir una nueva rutina, tendremos que poner otro número de línea en la instrucción **ON el GOSUB**. Editemos la línea 100:

```
100 ON el GOSUB 120,210,270,350,420
```

Para leer los datos tenemos que empezar por abrir (**OPEN**) en dirección de entrada (**IN**) el fichero que tenemos en la cinta con el nombre de '**datos**'. Después leemos con **INPUT** (por el canal **#9**) los valores de las variables **NOMBRE$(x)** y **TEL$(x)**, de 1 a 100, y finalmente cerramos (**CLOSE**) el fichero y retornamos (**INTRO**) al menú. Escriba las siguientes líneas:

```
420 OPENIN "!datos"
430 FOR x=1 TO 100
440 INPUT #9,NOMBRE$(x),TEL$(x)
450 NEXT
460 CLOSEIN
470 PRINT"Fin de la carga de los datos"
480 RETURN
```

El fin del principio

Por fin hemos conseguido un programa que realiza todas las funciones previstas. Todo lo que nos queda ahora es ocuparnos de cómo presentar los resultados en la pantalla.

El principio del fin

Para mejorar el manejo de la pantalla vamos a añadir unas cuantas instrucciones:

```
34 MODE 1
```

Esta instrucción establece el modo de pantalla y borra la pantalla al principio del programa. Añada ahora lo siguiente:

```
36 WINDOW #1,13,30,10,14
```

No se deje impresionar por esta instrucción aparentemente tan complicada. Lo que hemos hecho es definir una ventana en la pantalla para escribir en ella el menú. Después de la palabra **WINDOW**, lo primero que ponemos es el número del canal al que vamos a asignarla; recuerde que podemos elegir cualquiera entre el **0** y el **7**. Ahora bien, si tenemos en cuenta que todos los textos que enviamos a la pantalla sin especificar número de canal circulan por el **#0**, no vamos a dar este número a nuestra ventana. Elegimos, pues, el número **1**. Los cuatro números que vienen a continuación especifican el tamaño de la ventana. Su significado no podía ser más sencillo: columna del borde izquierdo, columna del derecho, primera fila y última fila, por este orden. En nuestro caso, después de especificar el canal **#1**, hemos dicho al ordenador que la ventana empieza en la columna número **13** y termina en la número **30**, en horizontal, y empieza en la fila **10** y termina en la **14**, en vertical.

Para dirigir los textos del menú al canal **#1** debemos editar las líneas 40 a 80:

```
40 PRINT #1,"1. Introd. datos"
50 PRINT #1,"2. Listado"
60 PRINT #1,"3. Busqueda"
70 PRINT #1,"4. Grabar datos"
80 PRINT #1,"5. Cargar datos"
```

Añadamos también

```
85 LOCATE 7,20
```

para situar el texto de la línea 90.

Para borrar la pantalla cada vez que el programa retorne al menú, modifiquemos la línea 110:

```
110 GOTO 34
```

Además, para borrar la pantalla cada vez que se selecciona una opción del menú añadimos:

```
95 CLS
```

Finalmente, incluiremos las tres líneas siguientes para hacer que el retorno al menú sea controlado por el usuario:

```
103 LOCATE 9,20
105 PRINT "Pulse una tecla para volver al menu"
107 IF INKEY$="" THEN 107
```

La línea 103 especifica dónde se debe escribir el mensaje de la línea 105. La línea 107 examina el teclado para averiguar si se ha pulsado alguna tecla. Si no se ha pulsado ninguna, **INKEY$** genera la cadena vacía y entonces se repite la ejecución de la línea hasta que efectivamente se pulse alguna. Ésta es la forma habitual de programar la espera 'hasta que el usuario se decida', pues BASIC no pasa a la línea siguiente mientras no se pulse una tecla.

Bueno, ya hemos terminado. O quizá no, porque podríamos incluir la posibilidad de modificar o borrar nombres y números de teléfono, de ordenarlos alfabéticamente, de listarlos por la impresora, etc. Todas estas mejoras son posibles; podríamos seguir mejorando y depurando el programa indefinidamente, pero en algún momento tenemos que poner el punto y aparte, y éste es tan bueno como cualquier otro. Le dejamos, pues, con este programa de 'guía de teléfonos', confiando en haberle abierto las puertas del arte de programar. Adecente el programa con la orden

```
RENUM
```

y grábelo en cinta; o bórrelo si no le gusta. Quién sabe; quizá pueda necesitarlo algún día.

Listado final:

```
10 REM guia de telefonos
20 DIM NOMBRE$(100)
30 DIM TEL$(100)
40 MODE 1
50 WINDOW #1,13,30,10,14
60 PRINT #1,"1. Introd. datos"
70 PRINT #1,"2. Listado"
80 PRINT #1,"3. Busqueda"
90 PRINT #1,"4. Grabar datos"
100 PRINT #1,"5. Cargar datos"
110 LOCATE 7,20
120 INPUT "Elija una opcion (1-5) ";el
130 CLS
140 ON el GOSUB 190,290,350,430,500
150 LOCATE 9,20
160 PRINT"Pulse una tecla para volver al menu"
170 IF INKEY$="" THEN 170
180 GOTO 40
190 FOR x=1 TO 100
200 CLS
210 IF LEN(NOMBRE$(x))>0 THEN 260
220 PRINT"Pulse [INTRO] para terminar"
230 INPUT;" Nombre ";NOMBRE$(x)
240 IF NOMBRE$(x)="" THEN 270
250 INPUT;" Telefono ";TEL$(x)
260 NEXT
270 PRINT "Fin de la entrada de datos"
```

```
280 RETURN
290 FOR x=1 TO 100
300 IF NOMBRE$(x)="" THEN 330
310 PRINT NOMBRE$(x);" ";TEL$(x)
320 NEXT
330 PRINT "Fin de la lista"
340 RETURN
350 INPUT "Buscar ";BUSCA$
360 FOR x=1 TO 100
370 IF INSTR(NOMBRE$(x),BUSCA$)=0 THEN 400
380 PRINT NOMBRE$(x);" ";TEL$(x)
390 END
400 NEXT
410 PRINT"No lo encuentro"
420 RETURN
430 OPENOUT "!datos"
440 FOR x=1 TO 100
450 WRITE #9,NOMBRE$(x),TEL$(x)
460 NEXT
470 CLOSEOUT
480 PRINT"Fin de la grabación de los datos"
490 RETURN
500 OPENIN "!datos"
510 FOR x=1 TO 100
520 INPUT #9,NOMBRE$(x),TEL$(x)
530 NEXT
540 CLOSEIN
550 PRINT"Fin de la carga de los datos"
560 RETURN
```

Capítulo 3
Lista completa de las palabras clave del BASIC de AMSTRAD

IMPORTANTE

Es imprescindible que usted comprenda la terminología y la notación utilizadas en este capítulo. Para describir la sintaxis de las diversas instrucciones utilizamos tanto paréntesis como corchetes. Es necesario distinguirlos, pues tienen significados diferentes.

Toda instrucción, o parte de instrucción, no encerrada entre corchetes debe ser escrita tal como aquí se muestra. Por ejemplo, la instrucción **END** tiene la forma:

END

Usted tiene que escribir la palabra **END** literalmente.

Cuando un texto está encerrado entre corchetes angulados, por ejemplo

⟨*número de línea*⟩

no hay que teclear ni los corchetes ni las palabras en ellos incluidas. El texto es una indicación del tipo de datos que se debe escribir. Por ejemplo,

EDIT ⟨*número de línea*⟩

significa que hay que escribir la palabra **EDIT** y a continuación un espacio y un número de línea:

EDIT 100

Los paréntesis ordinarios ( ) son obligatorios y han de ser tecleados explícitamente. Por ejemplo,

COS(⟨*expresión numérica*⟩**)**

exige que se escriban los paréntesis rodeando la expresión numérica cuyo coseno se quiera calcular:

PRINT COS(45)

Finalmente, los corchetes delimitan elementos opcionales. Por ejemplo,

RUN [⟨*número de línea*⟩]

significa que no es necesario poner un parámetro después de la palabra clave **RUN**, pero que sí se puede incluir ese parámetro; además, en caso de incluirlo, tiene que ser necesariamente un número de línea:

RUN o **RUN 100**

Caracteres especiales

& o **&H**	Prefijo para constante hexadecimal
&X	Prefijo para constante binaria
#	Prefijo para indicador de canal

Tipos de datos

Las cadenas literales pueden contener entre 0 y 255 caracteres. Una ⟨expresión literal⟩ es una expresión que toma como valor una cadena literal. Las cadenas literales se pueden concatenar (unir) con el operador +, a condición de que la cadena resultante no tenga más de 255 caracteres.

Los datos numéricos pueden ser enteros o reales. En este sentido, los enteros tienen la limitación adicional de estar comprendidos entre −32768 y +32767. El ordenador maneja números reales con precisión ligeramente superior a nueve dígitos. Los números reales están comprendidos en el margen ±1.7E+38. El menor valor mayor que cero es aproximadamente 2.9E−39.

Una ⟨expresión numérica⟩ es una expresión que toma como valor un número. Puede consistir en números concretos, en una variable numérica o en combinaciones de ambos: cualquier cosa que no sea una ⟨expresión literal⟩.

Una ⟨expresión de número de canal⟩ es una ⟨expresión numérica⟩ que identifica una ventana de la pantalla, la impresora o el sistema de cinta.

Una ⟨lista de:⟨elemento⟩ describe un parámetro que está formado por una lista de elementos separados por comas. Puede contener uno o varios elementos, sin más limitación que la longitud de la línea.

Los indicadores de tipo son:

%	Entero
!	Real (por defecto)
$	Literal

La lista de las palabras clave tiene la siguiente estructura:

PALABRA CLAVE
Sintaxis
Ejemplo
Descripción
Palabras clave asociadas

Las palabras clave pueden ser:

Órdenes: operaciones que se ejecutan directamente.
Funciones: operaciones que intervienen como argumentos de una expresión.
Operadores: operaciones que actúan sobre argumentos matemáticos.

BASIC convierte todas las palabras clave de minúsculas a mayúsculas cuando se lista el programa. En los ejemplos las mostraremos en mayúsculas. Es conveniente que usted las introduzca en minúsculas, pues entonces le será más fácil detectar los errores, dado que las palabras clave erróneamente tecleadas seguirán apareciendo en minúscula en los listados.

Lista de palabras clave

ABS

ABS(⟨*expresión numérica*⟩**)**

```
PRINT ABS(-67.98)
 67.98
```

Función. Calcula el valor absoluto de la expresión dada; es decir, convierte los números negativos en positivos.

Palabras clave asociadas: **SGN**.

AFTER

AFTER ⟨*retardo del temporizador*⟩[,⟨*número de temporizador*⟩] **GOSUB** ⟨*número de línea*⟩

```
10 AFTER 250 GOSUB 60:CLS
20 PRINT "Adivine una letra en 5 segundos"
30 a$=INKEY$:IF indicador=1 THEN END
40 IF a$<>CHR$(INT(RND*26+97)) THEN 30
50 PRINT a$;" es correcto. Usted gana."
55 SOUND 1,478:SOUND 1,358:END
60 PRINT "demasiado tarde. Yo gano."
70 SOUND 1,2000:indicador=1:RETURN
run
```

Orden. Invoca una subrutina de BASIC cuando ha transcurrido el tiempo especificado. El parámetro ⟨retardo del temporizador⟩ especifica el tiempo en unidades de 0.02 s (cincuentavos de segundo).

El ⟨número de temporizador⟩ (margen de 0 a 3) especifica cuál de los cuatro temporizadores disponibles debe medir el tiempo. El temporizador número 3 tiene la máxima prioridad; el 0 (que es el implícito), la mínima.

Cada temporizador puede tener una subrutina asociada.

Para más amplia información sobre las interrupciones, consulte la parte 2 del capítulo titulado 'Cuando usted guste . . .'.

Palabras clave asociadas: **EVERY**, **REMAIN**, **RETURN**.

AND

⟨*argumento*⟩ **AND** ⟨*argumento*⟩

```
IF "juan"<"pepe" AND "perro">"gato" THEN PRINT "verdadero" ELSE PRINT "falso"
verdadero
IF "pepe"<"juan" AND "gato">"perro" THEN PRINT "verdadero" ELSE PRINT "falso"
falso
IF "juan"<"pepe" AND "gato">"perro" THEN PRINT "verdadero" ELSE PRINT "falso"
falso
....
PRINT 1 AND 1
 1
PRINT 0 AND 0
 0
PRINT 1 AND 0
 0
```

Operador. Realiza la operación lógica 'y' bit a bit entre enteros. El resultado es **0** a menos que ambos bits sean **1**.

Para más amplia información sobre operaciones lógicas, consulte la parte 2 del capítulo titulado 'Cuando usted guste . . .' .

Palabras clave asociadas: **OR**, **NOT**, **XOR**.

ASC

ASC(⟨*expresión literal*⟩**)**

```
PRINT ASC("x")
 120
```

Función. Calcula el código ASCII del primer carácter de la ⟨expresión literal⟩.

Palabras clave asociadas: **CHR$**.

ATN

ATN(⟨*expresión numérica*⟩**)**

```
PRINT ATN(1)
 0.785398163
```

Función. Calcula el arco cuya tangente es la ⟨expresión numérica⟩.

Obsérvese que se puede utilizar **DEG** y **RAD** para hacer que el resultado se dé en grados o en radianes, respectivamente.

Palabras clave asociadas: **COS**, **DEG**, **RAD**, **SIN**, **TAN**.

AUTO

AUTO [⟨*número de línea*⟩][,⟨*incremento*⟩]

```
AUTO 100,50
```

Orden. Genera números de línea **AUTO**máticamente. El parámetro opcional ⟨número de línea⟩ especifica en qué número debe empezar la numeración. Si se lo omite, el ordenador genera números de línea a partir del 10.

El parámetro opcional ⟨incremento⟩ especifica el salto que debe haber entre cada línea y la siguiente. Si se lo omite, el ordenador genera números de línea espaciados de 10 en 10.

Si se genera un número de línea que ya ha sido utilizado, el ordenador informa del hecho mostrando un asterisco a la derecha del número.

Para interrumpir el proceso de numeración automática se pulsa [ESC].

Palabras clave asociadas: ninguna.

BIN$

BIN$(⟨*expresión entera*⟩[,⟨*expresión entera*⟩]**)**

```
PRINT BIN$(64,8)
 01000000
```

Función. Produce una cadena de dígitos binarios que representan el valor de la primera ⟨expresión entera⟩, utilizando el número de dígitos binarios especificados por la segunda ⟨expresión entera⟩ (margen de 0 a 16). Si el número de dígitos especificado es mayor que el necesario, la expresión resultante se completa con ceros por la izquierda; si el número de dígitos especificado es demasiado pequeño, la expresión resultante no se trunca, sino que contiene todos los dígitos que sean necesarios.

La ⟨expresión entera⟩ que se va a convertir en binario debe tener un valor comprendido entre −32768 y +65535.

Palabras clave asociadas: **HEX$**, **STR$**.

BORDER

BORDER ⟨*color*⟩[,⟨*color*⟩]

```
10 REM 729 combinaciones para el borde!
20 SPEED INK 5,5
30 FOR a=0 TO 26
40 FOR b=0 TO 26
50 BORDER a,b:CLS:LOCATE 14,13
60 PRINT "border";a;",";b
70 FOR t=1 TO 500
80 NEXT t,b,a
run
```

Orden. Cambia el color del borde de la pantalla. Si se especifican dos colores, el del borde alterna entre ambos, a una velocidad que se puede regular con la orden **SPEED INK**. El margen de colores es de 0 a 26.

Palabras clave asociadas: **SPEED INK**.

BREAK

Véase **ON BREAK GOSUB**, **ON BREAK STOP**.

CALL

CALL ⟨*expresión de dirección de memoria*⟩[,⟨*lista de:*⟨*parámetro*⟩]

CALL 0

Orden. Permite invocar desde BASIC una rutina escrita en código de máquina. El ejemplo anterior provoca la reinicialización de la máquina.

Esta orden no debe ser utilizada si no se sabe cuáles son sus efectos.

Palabras clave asociadas: **UNT**.

CAT

CAT

CAT

Orden. Hace que BASIC se ponga a examinar la cinta y dé una lista de los ficheros que va encontrando. Para cada fichero muestra cada uno de los bloques integrantes, con el siguiente formato:

NOMBREDEFICHERO Númerodebloque Indicador Ok

El **Indicador** puede ser:

$ para programas de BASIC no protegidos
% para programas de BASIC protegidos
* para ficheros ASCII
& para ficheros binarios

El mensaje **Ok** aparece al final de cada línea para indicar que el ordenador ha leído el fichero sin incidentes; es decir, que habría podido cargar el fichero de haberlo intentado.

Si un fichero ha sido grabado sin nombre, **CAT** escribe:

Unnamed file

Esta orden no afecta al programa actualmente residente en la memoria.

Palabras clave asociadas: **LOAD**, **RUN**, **SAVE**.

CHAIN

CHAIN ⟨*nombre de fichero*⟩[,⟨*expresión de número de línea*⟩]

CHAIN "programa",350

Orden. Lee un programa grabado en cinta y lo carga en la memoria reemplazando al actual. Inicia la ejecución del nuevo programa, bien desde el principio, bien desde una línea especificada por el parámetro opcional ⟨expresión de número de línea⟩.

Si no se especifica nombre de fichero, BASIC intenta cargar el primer fichero válido que encuentra en la cinta. Si el primer carácter del nombre es !, BASIC suprime tanto este carácter como todos los mensajes que de otra forma generaría el sistema de lectura de la cinta.

Los ficheros protegidos (grabados con **SAVE** ... **,p**) se pueden cargar y ejecutar con esta orden.

Palabras clave asociadas: **CHAIN MERGE**, **LOAD**, **MERGE**.

CHAIN MERGE

CHAIN MERGE ⟨*nombre de fichero*⟩[,⟨*expresión de número de línea*⟩]
[,**DELETE** ⟨*margen de números de línea*⟩]

CHAIN MERGE "nuevoprograma",750,DELETE 400–680

Orden. Lee un programa grabado en el disco, lo carga en la memoria mezclándolo con el actual y empieza a ejecutar el programa resultante, bien desde el principio, bien desde una línea especificada por el parámetro opcional ⟨expresión de número de línea⟩. Si se desea borrar parte del programa original antes de iniciar la carga del nuevo, se puede utilizar la opción **DELETE** ⟨margen de números de línea⟩.

Obsérvese que las líneas del programa que se carga sustituyen a las que tuvieran el mismo número en el programa original.

Si no se especifica nombre de fichero, BASIC intenta cargar el primer fichero válido que encuentra en la cinta. Si el primer carácter del nombre es !, BASIC suprime tanto este carácter como todos los mensajes que de otra forma generaría el sistema de lectura de la cinta.

CHAIN MERGE conserva los valores actuales de las variables, pero descarta las funciones definidas por el usuario y abandona los ficheros que estén abiertos. Además, desactiva la intercepción de errores que pueda haber sido activada con **ON ERROR GOTO**, ejecuta implícitamente una orden **RESTORE**, reinicializa todas las características establecidas con **DEFINT**, **DEFREAL** y **DEFSTR** y abandona las instrucciones **FOR**, **WHILE** y **GOSUB** que estén activas.

Los ficheros protegidos (grabados con **SAVE** ... **,p**) no se pueden cargar con esta orden.

Palabras clave asociadas: **CHAIN**, **DELETE**, **LOAD**, **MERGE**.

CHR$

CHR$(⟨*expresión entera*⟩**)**

```
10 FOR x=32 TO 255
20 PRINT x;CHR$(x),
30 NEXT
run
```

Función. Convierte la ⟨expresión entera⟩, que tiene que estar en el margen de 0 a 255, en su carácter equivalente. El conjunto de caracteres del AMSTRAD se muestra en la parte 3 del capítulo titulado 'Para su referencia ...'.

Obsérvese que los números 0 a 31 generan caracteres de control; por esa razón los hemos excluido del ejemplo anterior.

Palabras clave asociadas: **ASC**, **LEFT$**, **MID$**, **RIGHT$**, **STR$**.

CINT

CINT(⟨*expresión numérica*⟩)

```
10 n=1.9999
20 PRINT CINT(n)
run
 2
```

Función. Aproxima el valor de la ⟨expresión numérica⟩ a un número entero comprendido en el margen de −32768 a +32767.

Palabras clave asociadas: **CREAL**, **FIX**, **INT**, **ROUND**, **UNT**.

CLEAR

CLEAR

```
CLEAR
```

Orden. Borra todas las variables (iguala a cero las numéricas y a la cadena vacía las literales). Además, se abandonan los ficheros que estén abiertos, se borran todas las listas y funciones definidas por el usuario y BASIC retorna al modo de radianes para el cálculo de funciones trigonométricas.

CLG

CLG [⟨*tinta*⟩]

```
LOCATE 1,20
CLG 3
```

Orden. Borra la pantalla gráfica, dejándola del color del papel de gráficos. Si se especifica la ⟨tinta⟩, el papel de gráficos toma ese color.

Palabras clave asociadas: **CLS**, **INK**, **ORIGIN**.

CLOSEIN

CLOSEIN

```
CLOSEIN
```

Orden. Cierra el fichero que esté abierto en dirección de entrada (si lo hay). (Véase **OPENIN**.)

Palabras clave asociadas: **EOF**, **OPENIN**.

CLOSEOUT

CLOSEOUT

```
CLOSEOUT
```

Orden. Cierra el fichero que esté abierto en dirección de salida (si lo hay). (Véase **OPENOUT**.)

Palabras clave asociadas: **OPENOUT**.

CLS

CLS [#⟨*expresión de número de canal*⟩]

```
10 PAPER #2,3
20 CLS #2
run
```

Orden. Borra la ventana (canal de pantalla) especificada, tiñéndola del color del papel. Si no se especifica ninguna ⟨expresión de número de canal⟩, el ordenador borra la pantalla **#0**.

Palabras clave asociadas: **CLG**, **INK**, **PAPER**, **WINDOW**.

CONT

CONT

```
CONT
```

Orden. Reanuda la ejecución del programa cuando éste ha sido interrumpido, bien porque se ha pulsado la tecla [ESC] dos veces, bien porque el programa ha encontrado la instrucción **STOP**. **CONT** solamente puede continuar el programa en el supuesto de que no se lo haya modificado durante la interrupción y de que no se trate de un programa protegido.

Palabras clave asociadas: **STOP**.

COS

COS(⟨*expresión numérica*⟩**)**

```
DEG
PRINT COS(45)
 0.707106781
```

Función. Calcula el coseno de la ⟨expresión numérica⟩.

Obsérvese que se puede utilizar **DEG** y **RAD** para hacer que el ordenador entienda los arcos como dados en grados o en radianes, respectivamente.

Palabras clave asociadas: **ATN**, **DEG**, **RAD**, **SIN**, **TAN**.

CREAL

CREAL(⟨*expresión numérica*⟩**)**

```
10 a=PI
20 PRINT CINT(a)
30 PRINT CREAL(a)
run
 3
 3.14159265
```

Función. Convierte el valor de la ⟨expresión numérica⟩, cualquiera que sea su tipo, en un número real.

Palabras clave asociadas: **CINT**.

DATA

DATA ⟨*lista de:*⟨*constante*⟩

```
10 FOR x=1 TO 4
20 READ nombre$,apellido$
30 PRINT "Don ";nombre$;" ";apellido$
40 NEXT
50 DATA Manuel,Gonzalez,Daniel,Garcia
60 DATA Felipe,Revilla,Ernesto,Diaz
run
```

Orden. Identifica las líneas en las que están contenidos los datos que pueden ser leídos con la instrucción **READ**. Cada vez que se lee un dato, el 'puntero' pasa a señalar el siguiente elemento de la lista. Se puede utilizar la instrucción **RESTORE** para hacer que el puntero señale una línea de datos determinada.

En la parte 2 del capítulo titulado 'Cuando usted guste . . .' daremos más amplia información sobre los datos de este tipo.

Palabras clave asociadas: **READ**, **RESTORE**.

DEF FN

DEF FN⟨*nombre de la función*⟩[(⟨*parámetros formales*⟩)]=⟨*expresión*⟩

```
10 t=TIME/300
20 DEF FNreloj=INT(TIME/300-t)
30 EVERY 100 GOSUB 50
40 GOTO 40
50 PRINT "El programa se ejecuto hace";
60 PRINT FNreloj;"segundos"
70 RETURN
run
```

Orden. BASIC permite que el programa defina y utilice funciones. **DEF FN** realiza la primera parte de ese mecanismo: crea una función específica del programa; una vez definida, la función se maneja de la misma forma que las funciones intrínsecas de BASIC.

(Nótese en el ejemplo anterior que el valor de la función **FNreloj** sigue siendo actualizada ininterrumpidamente, aunque se imponga una pausa al programa pulsando [ESC] dos veces para luego reanudarlo con **CONT**.)

DEFINT

DEFINT ⟨*lista de:⟨margen de letras*⟩

```
10 DEFINT n
20 numero=123.456
30 PRINT numero
run
 123
```

Orden. Cada vez que el programa encuentra una variable que no vaya seguida de un indicador explícito de tipo (! % $), el ordenador considera que es del tipo implícito. Esta orden hace que el tipo implícito sea 'variable entera'. La orden afecta a todas las variables cuyos nombres empiecen por la letra o letras especificadas. Se puede especificar una lista:

```
DEFINT a,b,c
```

o bien un margen de iniciales:

```
DEFINT a-z
```

Palabras clave asociadas: **DEFREAL**, **DEFSTR**.

DEFREAL

DEFREAL ⟨*lista de:⟨margen de letras*⟩

```
DEFREAL x,a-f
```

Orden. Cada vez que el programa encuentra una variable que no vaya seguida de un indicador explícito de tipo (! % $), el ordenador considera que es del tipo implícito. Esta orden hace que el tipo implícito sea 'variable real'. La orden afecta a todas las variables cuyos nombres empiecen por la letra o letras especificadas. Se puede especificar una lista:

```
DEFREAL a,b,c
```

o bien un margen de iniciales:

```
DEFREAL a-z
```

Palabras clave asociadas: **DEFINT**, **DEFSTR**.

DEFSTR

DEFSTR ⟨*lista de:⟨margen de letras*⟩

```
10 DEFSTR n
20 nombre="Amstrad"
30 PRINT nombre
run
Amstrad
```

Orden. Cada vez que el programa encuentra una variable que no vaya seguida de un indicador explícito de tipo (! % $), el ordenador considera que es del tipo implícito. Esta orden hace que el tipo implícito sea 'variable literal'. La orden afecta a todas las variables cuyos nombres empiecen por la letra o letras especificadas. Se puede especificar una lista:

```
DEFSTR a,b,c
```

o bien un margen de iniciales:

```
DEFSTR a-z
```

Palabras clave asociadas: **DEFINT**, **DEFREAL**.

DEG

DEG

```
DEG
```

Orden. Hace que BASIC considere que los valores de los arcos que intervienen en los cálculos trigonométricos están expresados en grados, no en radianes. La situación implícita para las funciones **SIN**, **COS**, **TAN** y **ATN** es la inversa: radianes. **DEG** cambia de radianes a grados; las órdenes **RAD**, **NEW**, **CLEAR**, **LOAD**, **RUN**, etc. cambian de grados a radianes.

Palabras clave asociadas: **ATN**, **COS**, **RAD**, **SIN**, **TAN**.

DELETE

DELETE ⟨*margen de números de línea*⟩

```
DELETE 100-200
```

Orden. Borra la instrucción o grupo de instrucciones especificado por ⟨margen de números de línea⟩.

Si se omite el primer número de ⟨margen de números de línea⟩, se entiende 'desde el principio del programa hasta . . .'; si se omite el segundo número, se entiende '. . . hasta el final del programa'. Así,

```
DELETE -200
```

borra las primeras líneas del programa, hasta la 200 inclusive. En cambio,

```
DELETE 50-
```

borra desde la línea 50 en adelante. Finalmente,

```
DELETE
```

borra el programa completo.

Palabras clave asociadas: **CHAIN, MERGE, NEW, RENUM.**

DI

DI

```
10 CLS:TAG:EVERY 10 GOSUB 90
20 X1=RND*320:X2=RND*320
30 Y=200+RND*200:C$=CHR$(RND*255)
40 FOR X=320-X1 TO 320+X2 STEP 4
50 DI
60 PLOT 320,0,1:MOVE X-2,Y:MOVE X,Y
70 PRINT " ";C$;:CALL &BD19
80 EI:NEXT:GOTO 20
90 MOVE 320,0:DRAW X+8,Y-16,0:RETURN
run
```

Orden. Inhibe interrupciones (a excepción de [ESC]) hasta que se las habilite explícitamente con **EI** o implícitamente con el **RETURN** en el que termina la subrutina invocada como consecuencia de la interrupción.

Nótese que al entrar en una rutina de interrupción se inhiben automáticamente todas las interrupciones de prioridad igual o inferior.

Esta orden se utiliza para asegurar que el programa funcione sin interrupción; por ejemplo, cuando dos rutinas están compitiendo por el uso de recursos comunes. En el ejemplo anterior, el programa principal y la subrutina de interrupción están compitiendo por el uso de la pantalla gráfica.

En la parte 2 del capítulo 'Cuando usted guste . . .' daremos más amplia información a este respecto.

Palabras clave asociadas: **AFTER, EI, EVERY, REMAIN.**

DIM

DIM ⟨*lista de:⟨variable indexada⟩*

```
10 CLS
20 DIM amigo$(5),telefono$(5)
30 FOR n=1 TO 5
40 PRINT "Amigo numero:";n
50 INPUT "Nombre ";amigo$(n)
60 INPUT "Telefono ";telefono$(n)
70 PRINT
80 NEXT
90 FOR n=1 TO 5
100 PRINT n;amigo$(n),telefono$(n)
110 NEXT
run
```

Orden. DIMensiona listas. **DIM** hace una reserva de espacio para los elementos de la lista y especifica cuál va a ser el máximo valor del índice. BASIC requiere que se le informe del tamaño que debe reservar para las listas; sin embargo, permite el manejo de listas no dimensionadas (es decir, no declaradas con una orden **DIM**) a condición de que el índice no sobrepase el valor 10.

Una lista es una 'variable indexada', lo que quiere decir que se utiliza un solo nombre de variable para representar varios elementos; cada elemento está identificado por su 'índice'. Las listas pueden ser controladas fácilmente mediante bucles de tipo **FOR** ... **NEXT**, identificando el índice con la variable del contador del bucle.

Obsérvese que el valor mínimo del índice (esto es, el primer elemento utilizable) es el cero.

Un *vector* es una lista unidimensional. También se pueden utilizar listas multidimensionales (*matrices*). Por ejemplo, una lista tridimensional se dimensionaría con

```
DIM posicion$(20,20,20)
```

y un elemento concreto sería **posicion$(4,5,6)**.

Palabras clave asociadas: **ERASE**.

DRAW

DRAW ⟨*coordenada x*⟩,⟨*coordenada y*⟩[,⟨*tinta*⟩]

```
10 MODE 0:BORDER 0:PAPER 0:INK 0,0
20 x=RND*640:y=RND*400:z=RND*15
30 DRAW x,y,z
40 GOTO 20
run
```

Orden. Dibuja en la pantalla gráfica una recta que va desde la posición actual del cursor gráfico hasta el punto de coordenadas *x, y* especificadas. Se puede especificar el color de ⟨tinta⟩ con el que se desea dibujar la recta (margen: **0** a **15**).

Palabras clave asociadas: **DRAWR, MOVE, MOVER, ORIGIN, PLOT, PLOTR, TEST, TESTR, XPOS, YPOS.**

DRAWR

DRAWR ⟨*incremento de x*⟩,⟨*incremento de y*⟩[,[⟨*tinta*⟩][,⟨*modo de tinta*⟩]]

```
10 CLS:PRINT "escalera"
20 MOVE 0,350:FOR n=1 TO 8
30 DRAWR 50,0
40 DRAWR 0,-50
50 NEXT
60 GOTO 60
run
```

Orden. Dibuja en la pantalla gráfica una recta que va desde la posición actual del cursor gráfico hasta el punto especificado en coordenadas relativas por ⟨incremento de x⟩ e ⟨incremento de y⟩. Se puede especificar el color de ⟨tinta⟩ con el que se desea dibujar (margen: **0** a **15**).

Palabras clave asociadas: **DRAW, MOVE, MOVER, ORIGIN, PLOT, PLOTR, TEST, TESTR, XPOS, YPOS.**

EDIT

EDIT ⟨*número de línea*⟩

```
EDIT 20
```

Orden. Escribe en la pantalla la línea especificada por ⟨número de línea⟩ y la deja dispuesta para ser editada, con el cursor sobre ella.

Palabras clave asociadas: **LIST.**

EI

EI

```
EI
```

Orden. Habilita interrupciones que hayan sido inhibidas previamente con **DI.**

Si se inhiben las interrupciones en una subrutina de interrupción, BASIC las habilita cuando encuentra el **RETURN** del final de la subrutina.

En la parte 2 del capítulo titulado 'Cuando usted guste . . .' daremos más amplia información acerca de las interrupciones.

Palabras clave asociadas: **AFTER, DI, EVERY, REMAIN.**

ELSE

(Véase **IF.**)

END

END

```
END
```

Orden. Termina la ejecución del programa y devuelve el ordenador al modo directo. En un mismo programa se pueden incluir varias órdenes **END**; se supone un **END** implícito después de la última línea del programa.

Palabras clave asociadas: **STOP.**

ENT

ENT ⟨*número de envolvente*⟩[,⟨*sección de envolvente*⟩][,⟨*sección de envolvente*⟩]
[,⟨*sección de envolvente*⟩][,⟨*sección de envolvente*⟩][,⟨*sección de envolvente*⟩]

```
10 ENT 1,10,-50,10,10,50,10
20 SOUND 1,500,200,10,,1
run
```

Orden. Define las características de la envolvente de tono especificada por ⟨número de envolvente⟩ (margen: de **1** a **15**) y citada como ⟨envolvente de tono⟩ en la orden **SOUND**. Si el ⟨número de envolvente⟩ es negativo (margen de −1 a −15), la envolvente se repite hasta el final de la ⟨duración⟩ especificada en la orden **SOUND**.

Cada ⟨sección de envolvente⟩ puede constar de 2 o 3 parámetros.

Si se utilizan 3 parámetros, tienen el significado siguiente:

⟨*número de escalones*⟩,⟨*altura de cada escalón*⟩,⟨*tiempo de pausa*⟩

Parámetro 1: ⟨número de escalones⟩

Este parámetro especifica en cuántas etapas queremos que la nota evolucione a lo largo de la duración de la sección de la envolvente. Por ejemplo, si una sección de la nota ha de durar 10 segundos, se la puede dividir en 10 etapas de 1 segundo cada una; en tal caso, el parámetro ⟨número de escalones⟩ tendrá que ser 10.

El margen de este parámetro es de **0** a **239**.

Parámetro 2: ⟨altura de cada escalón⟩

En cada etapa, el tono puede variar con respecto al de la anterior en un número de unidades comprendido entre − 128 y + 127. Las variaciones negativas representan aumentos del tono (tono más agudo); las variaciones positivas reducen el tono (tono más grave). El valor mínimo del período de tono es 0. En el capítulo 'Para su referencia . . .' se da la lista completa de los periodos de tono.

Parámetro 3: ⟨tiempo de pausa⟩

Este parámetro especifica la duración de cada escalón (intervalo entre cada dos saltos) en unidades de centésimas de segundo. El margen de valores es de **0** a **255**; esto significa que la duración máxima de cada escalón es de 2.56 segundos (el **0** se considera como **256**).

Si se utilizan 2 parámetros, su significado es el siguiente:

⟨*periodo de tono*⟩,⟨*tiempo de pausa*⟩

Parámetro 1: ⟨periodo de tono⟩

Este parámetro da un nuevo valor absoluto del periodo de tono. (Véase el parámetro 2 de la orden **SOUND**.)

Parámetro 2: ⟨tiempo de pausa⟩

Este parámetro especifica la duración de escalón en unidades de centésimas de segundo. El margen de valores es de **0** a **255**, es decir, el valor máximo es 2.56 segundos, ya que el **0** se considera como **256**.

General

La suma de todos los ⟨tiempos de pausa⟩ no debe ser mayor que el parámetro ⟨duración⟩ de la orden **SOUND**; de lo contrario, el sonido habrá concluido antes de que se hayan completado todas las etapas de la envolvente. (En tal caso, el ordenador ignora el resto del contenido de la envolvente.)

Análogamente, si la duración especificada en **SOUND** es mayor que la impuesta por la suma de todos los ⟨tiempos de pausa⟩, la nota continuará sonando aunque se hayan terminado los escalones de la envolvente de tono, y lo hará con el tono correspondiente al de la última etapa de la envolvente.

En una orden **ENT** se puede definir un máximo de 5 ⟨secciones de envolvente⟩ (cada una de las cuales puede constar de 2 o 3 parámetros).

La primera etapa de una envolvente de tono se ejecuta inmediatamente.

Cada vez que se define una envolvente de tono, se anula la definición anterior.

Si se define un ⟨número de envolvente⟩ sin especificar parámetros para ninguna de sus secciones, se anula la definición anterior.

En la parte 2 del capítulo 'Cuando usted guste . . .' daremos más información sobre las envolventes de tono.

Palabras clave asociadas: **ENV**, **SOUND**.

ENV

ENV ⟨*número de envolvente*⟩[,⟨*sección de envolvente*⟩][,⟨*sección de envolvente*⟩]
[,⟨*sección de envolvente*⟩][,⟨*sección de envolvente*⟩][,⟨*sección de envolvente*⟩]

```
10 ENV 1,15,-1,10,15,1,10
20 SOUND 1,200,300,15,1
run
```

Orden. Define las características de la envolvente de volumen especificada por ⟨número de envolvente⟩ (margen: de **1** a **15**) y citada como ⟨envolvente de volumen⟩ en la orden **SOUND**.

Cada ⟨sección de envolvente⟩ puede constar de 2 o 3 parámetros.

Si se utilizan 3 parámetros, tienen el significado siguiente:

⟨*número de escalones*⟩,⟨*altura de cada escalón*⟩,⟨*tiempo de pausa*⟩

Parámetro 1: ⟨número de escalones⟩

Este parámetro especifica en cuántas etapas queremos que la nota evolucione antes de completar la sección de la envolvente. Por ejemplo, si una sección de la nota ha de durar 10 segundos, se la puede dividir en 10 etapas de 1 segundo cada una; en tal caso, el parámetro ⟨número de escalones⟩ será **10**.

El margen de este parámetro es de **0** a **127**.

Parámetro 2: ⟨altura de cada escalón⟩

En cada etapa, el volumen puede variar con respecto al nivel anterior en un número de unidades comprendido entre **0** y **15**. Estos 16 niveles de sonido son los mismos que se definen en la instrucción **SOUND**. Sin embargo, el margen del parámetro ⟨altura de cada escalón⟩ es de **−128** a **+127**; el nivel del volumen vuelve a **0** cada vez que sobrepasa el **15**.

Parámetro 3: ⟨tiempo de pausa⟩

Este parámetro especifica la duración de cada escalón en unidades de centésimas de segundo. El margen de valores es de **0** a **255**; esto significa que la duración máxima de cada escalón es de 2.56 segundos (el **0** se considera como **256**).

Si se utilizan 2 parámetros, su significado es el siguiente:

⟨*envolvente de hardware*⟩,⟨*periodo de envolvente*⟩

Parámetro 1: ⟨envolvente de hardware⟩

Este parámetro especifica el valor que se debe enviar al registro de forma de envolventes del generador de sonido.

Parámetro 2: ⟨periodo de envolvente⟩

Este parámetro especifica el valor que se debe enviar a los registros de periodo de envolventes del generador de sonido.

Para utilizar las envolventes de hardware es necesario que el usuario conozca el funcionamiento interno del generador de sonido. De no ser así, le recomendamos que utilice las envolventes definidas por programa, incorporado un ⟨tiempo de pausa⟩ adecuado.

General

La suma de todos los ⟨tiempos de pausa⟩ no debe ser mayor que el parámetro ⟨duración⟩ de la orden **SOUND**; de lo contrario, el sonido habrá concluido antes de que se hayan completado todas las etapas de la envolvente. (En tal caso, el ordenador ignora el resto del contenido de la envolvente.)

Análogamente, si la duración especificada en **SOUND** es mayor que la impuesta por la suma de todos los ⟨tiempos de pausa⟩, la nota continuará sonando aunque se hayan terminado los escalones de la envolvente de tono, y lo hará con nivel de volumen igual al de la última etapa de la envolvente.

En una orden **ENV** se puede definir un máximo de 5 ⟨secciones de envolvente⟩ (cada una de las cuales puede constar de 2 o 3 parámetros).

La primera etapa de una envolvente de volumen se ejecuta inmediatamente.

Cada vez que se define una envolvente de volumen, se anula la definición anterior.

Si se define un ⟨número de envolvente⟩ sin especificar parámetros para ninguna de sus secciones, se anula la definición anterior.

En la parte 2 del capítulo 'Cuando usted guste . . .' daremos más información sobre las envolventes de volumen.

Palabras clave asociadas: **ENT**, **SOUND**.

EOF

EOF

```
10 OPENIN "ejemplo"
20 WHILE NOT EOF
30 LINE INPUT #9,a$
40 PRINT a$
50 WEND
60 CLOSEIN
run
```

Función. Comprueba si se ha alcanzado el final del fichero (End of File). Da el valor −1 (verdadero) si no hay ningún fichero abierto o si se ha alcanzado el final del fichero; da 0 (falso) en caso contrario.

Palabras clave asociadas: **OPENIN**, **CLOSEIN**.

ERASE

ERASE ⟨*lista de:⟨nombre de variable*⟩

```
DIM a(100),b$(100)
ERASE a,b$
```

Orden. Borra el contenido de una o varias listas y libera el espacio de memoria ocupado por ellas.

Palabras clave asociadas: **DIM**.

ERL

ERL

```
10 ON ERROR GOTO 30
20 GOTO 1000
30 PRINT "Error en la linea";ERL
40 END
run
```

Función. Da el número de la línea en la que se ha detectado el último error. En el programa ejemplo, el error está en la línea 20, y éste es el número generado por **ERL** en la línea 30.

Palabras clave asociadas: **ERR**, **ERROR**, **ON ERROR GOTO**, **RESUME**.

ERR

ERR

```
GOTO 500
Line does not exist
Ready
PRINT ERR
 8
```

Función. Da el número del último error detectado. En el capítulo titulado 'Para su referencia . . .' se da la lista de los mensajes de error. El programa ejempo muestra que el error número 8 es **'Line does not exist'** ("no existe esa línea").

Palabras clave asociadas: **ERL, ERROR, ON ERROR GOTO, RESUME.**

ERROR

ERROR ⟨*expresión entera*⟩

```
10 IF INKEY$=" " THEN 10 ELSE ERROR 17
run
```

Orden. Invoca el error de número especificado por ⟨expresión entera⟩. En el capítulo 'Para su referencia . . .' se da la lista de los mensajes de error 1 a 32. BASIC trata el **ERROR** como si efectivamente lo hubiera detectado: salta a la rutina de gestión de errores (si la hay) y genera los valores correctos de **ERR** y **ERL.**

La orden **ERROR** seguida de una ⟨expresión entera⟩ cuyo valor esté entre 33 y 255 sirve para crear mensajes de error a medida de las necesidades del programador, como ilustra el siguiente ejemplo:

```
10 ON ERROR GOTO 100
20 INPUT "Escriba una letra ";a$
30 IF LEN(A$)<>1 THEN ERROR 100
40 GOTO 20
100 IF ERR=100 THEN 110 ELSE 130
110 PRINT CHR$(7)
120 PRINT "Le dije que UNA letra!"
130 RESUME 20
run
```

Palabras clave asociadas: **ERL, ERR, ON ERROR GOTO, RESUME.**

EVERY

EVERY ⟨*periodo de tiempo*⟩[,⟨*número de temporizador*⟩] **GOSUB** ⟨*número de línea*⟩

```
10 EVERY 50,1 GOSUB 30
20 GOTO 20
30 SOUND 1,20
40 RETURN
run
```

Orden. Invoca una subrutina de BASIC a intervalos de tiempo regulares. El ⟨periodo de tiempo⟩ especifica el intervalo en unidades de 0.02 s (cintuentavos de segundo).

El ⟨número del temporizador⟩ (margen de **0** a **3**) especifica cuál de los cuatro cronómetros se debe utilizar. El temporizador número 3 tiene la prioridad más alta; el número **0** (que es el implícito), la más baja.

Cada uno de los cuatro cronómetros puede tener una subrutina asociada.

En la parte 2 del capítulo 'Cuando usted guste . . .' daremos más amplia información acerca de las interrupciones.

Palabras clave asociadas: **AFTER**, **REMAIN**.

EXP

EXP(⟨*expresión numérica*⟩**)**

```
PRINT EXP(6.876)
 968.743625
```

Función. Calcula el número *e* elevado a la potencia dada por ⟨expresión numérica⟩. El número *e* es la base de los logaritmos neperianos, aproximadamente igual a 2.7182818.

Palabras clave asociadas: **LOG**.

FIX

FIX(⟨*expresión numérica*⟩**)**

```
PRINT FIX(9.99999)
 9
```

Función. Da la parte entera de la ⟨expresión numérica⟩, ignorando su parte decimal.

Palabras clave asociadas: **CINT**, **INT**, **ROUND**.

FN

Véase **DEF FN**.

FOR

FOR ⟨*variable sencilla*⟩=⟨*principio*⟩ **TO** ⟨*fin*⟩ [**STEP** ⟨*salto*⟩]

```
10 FOR n=2 TO 8 STEP 2
20 PRINT n;
30 NEXT n
40 PRINT ", y ocho dieciseis"
run
```

Orden. Ejecuta el grupo de instrucciones comprendidas entre **FOR** y **NEXT** un número dado de veces, incrementando el valor de la variable de control (variable del contador) en un número especificado por ⟨salto⟩ desde ⟨principio⟩ hasta ⟨fin⟩. Si no se especifica ningún valor de ⟨salto⟩ para **STEP**, se supone el 1.

El ⟨salto⟩ puede ser una ⟨expresión numérica⟩. Si su valor es negativo, el parámetro ⟨principio⟩ tiene que ser mayor que el parámetro ⟨fin⟩, pues de no ser así la variable del contador no será modificada.

Los bucles **FOR ... NEXT** pueden ser 'anidados', es decir, programados uno dentro de otro.

No es necesario especificar el nombre de la variable a continuación de **NEXT**, ya que BASIC determina automáticamente a qué **FOR** corresponde cada **NEXT** 'anónimo'.

Palabras clave asociadas: **NEXT**, **STEP**, **TO**, **WHILE**.

FRE

FRE(⟨*expresión numérica*⟩**)**
FRE(⟨*expresión literal*⟩**)**

```
PRINT FRE(0)
PRINT FRE("")
```

Función. Da el espacio libre de memoria utilizable por BASIC. En la forma **FRE ("")** realiza una 'limpieza de memoria' antes de dar el número de bytes libres.

Palabras clave asociadas: **HIMEM**, **MEMORY**.

GOSUB

GOSUB ⟨*número de línea*⟩

```
GOSUB 210
```

Orden. Invoca una subrutina de BASIC; es decir, reconduce el programa hacia el ⟨número de línea⟩ especificado. El final de la subrutina estará indicado por la orden **RETURN**; cuando se llega a ella, el programa retorna a la instrucción siguiente al **GOSUB** que invocó la subrutina.

Palabras clave asociadas: **RETURN**.

GOTO

GOTO ⟨*número de línea*⟩

```
GOTO 90
```

Orden. Hace que la ejecución del programa salte a la línea especificada.

HEX$

HEX$(⟨*expresión entera*⟩[,⟨*anchura del campo*⟩]**)**

```
PRINT HEX$(255,4)
00FF
```

Función. Produce una cadena literal que es la versión hexadecimal de la ⟨expresión entera⟩. La longitud de la cadena es la especificada por el parámetro ⟨anchura del campo⟩ (margen de **0** a **16**). Si la longitud es demasiado pequeña, la cadena resultante no se trunca, sino que consta de todos los caracteres necesarios.

El valor de la ⟨expresión entera⟩ tiene que estar comprendido entre −32768 y +65535.

Palabras clave asociadas: **BIN$**, **STR$**, **UNT**.

HIMEM

HIMEM

```
PRINT HIMEM
 43903
```

Función. Da la dirección de la posición de memoria más alta accesible a BASIC (la cual puede ser modificada con la orden **MEMORY**).

Palabras clave asociadas: **FRE, MEMORY, SYMBOL, SYMBOL AFTER.**

IF

IF ⟨*expresión lógica*⟩ **THEN** ⟨*opción*⟩ [**ELSE** ⟨*opción*⟩]

```
10 MODE 1
20 x=CINT(RND*100)
30 PRINT "Adivine mi numero (de 0 a 100)"
40 INPUT n
50 IF n<x THEN PRINT n;"es demasiado bajo ..."
60 IF n>x THEN PRINT n;"es demasiado alto ..."
70 IF n=x THEN 80 ELSE c=c+1:GOTO 40
80 PRINT "Muy bien. Lo ha conseguido en";
90 PRINT c+1;"intentos!"
run
```

Orden. Determina si la ⟨expresión logica⟩ tiene el valor 'verdadero'. Si es así, ejecuta la primera ⟨opción⟩. Si la ⟨expresión lógica⟩ tiene el valor 'falso', ejecuta la ⟨opción⟩ especificada después de **ELSE**; si no se ha incluido **ELSE**, BASIC pasa a la siguiente línea del programa.

Las órdenes **IF** ... **THEN** ... se pueden anidar, sin más límite que la longitud de la línea. La orden (o la serie de órdenes anidadas) termina en el final de la línea, de modo que no es posible poner en la misma línea instrucciones que sean independientes de la orden **IF** ... **THEN** ... .

Si la ⟨opción⟩ especificada consiste en un salto de línea, la orden puede tener cualquiera de las tres formas siguientes:

```
IF a=1 THEN 100
IF a=1 GOTO 100
IF a=1 THEN GOTO 100
```

Palabras clave asociadas: **ELSE, GOTO, THEN.**

INK

INK ⟨*tinta*⟩,⟨*color*⟩[,⟨*color*⟩]

```
10 MODE 1:PAPER 0:PEN 1
20 FOR p=0 TO 1
```

```
30 FOR i=0 TO 26
40 INK p,i
50 LOCATE 16,12:PRINT "ink";p;",";i
60 FOR t=1 TO 400:NEXT t,i,p
70 INK 0,1:INK 1,24:CLS
run
```

Orden. Asigna uno o dos colores a la ⟨tinta⟩ dada. El parámetro ⟨tinta⟩ es el número de referencia de la tinta (margen de **0** a **15**), por el que se la cita en las órdenes **PEN** y **PAPER**. El primer ⟨color⟩ debe ser una expresión entera cuyo valor esté comprendido entre 0 y 26. Si se incluye el segundo ⟨color⟩, la tinta alterna entre los dos colores, a una velocidad que se controla con la orden **SPEED INK**.

Palabras clave asociadas: **PAPER**, **PEN**, **SPEED INK**.

INKEY

INKEY(⟨*expresión entera*⟩**)**

```
10 CLS:IF INKEY(55)=32 THEN 30 ELSE 20
20 CLS:GOTO 10
30 PRINT "Ha pulsado [MAYS] y V"
40 FOR t=1 TO 1000:NEXT:GOTO 10
run
```

Función. Examina el teclado para determinar si se ha pulsado una tecla determinada. La exploración se realiza una vez cada 0.02 s.

Esta función es útil para averiguar si se ha pulsado cierta tecla, comprobando si el número generado por **INKEY** es −1 (el cual es independiente de la situación de [MAYS] y [CTRL]).

El ejemplo anterior detecta cuándo se pulsa **V** (tecla número 55) con [MAYS] y sin [CTRL]. El esquema de la numeración de las teclas está grabado en la carcasa del ordenador, a la derecha del teclado, e impreso en el capítulo titulado 'Para su referencia . . .'.

Los valores generados por **INKEY** en función de la situación de las teclas [MAYS] y [CTRL] son los siguientes:

Valor generado	[MAYS]	[CTRL]	Tecla especificada
−1	Indiferente	Indiferente	No pulsada
0	No pulsada	No pulsada	Pulsada
32	Pulsada	No pulsada	Pulsada
128	No pulsada	Pulsada	Pulsada
160	Pulsada	Pulsada	Pulsada

Palabras clave asociadas: **INKEY$**, **INPUT**, **JOY**.

INKEY$

INKEY$

```
10 CLS
20 PRINT "Elija Si o No (S/N):"
30 a$=INKEY$
40 IF a$="" THEN 30
50 IF a$="s" OR a$="S" THEN 80
60 IF a$="n" OR a$="N" THEN 90
70 GOTO 30
80 PRINT "Ha respondido SI":END
90 PRINT "Ha respondido NO"
run
```

Función. Examina el teclado para determinar si se ha pulsado alguna tecla; de ser así, genera una cadena literal consistente en el carácter correspondiente a la tecla pulsada. Si no se ha pulsado ninguna tecla, genera la cadena vacía. En el programa ejemplo, las líneas 30 y 40 forman un bucle que no deja de repetirse mientras no se pulse una tecla.

Palabras clave asociadas: **INKEY**, **INPUT**.

INP

INP(⟨*número de puerta*⟩**)**

```
PRINT INP(&FF77)
 255
```

Función. Da el valor de entrada leído en la dirección de entrada/salida especificada por el parámetro ⟨número de puerta⟩.

Palabras clave asociadas: **OUT**, **WAIT**.

INPUT

INPUT [#⟨*expresión de número de canal*⟩,][;]
[⟨*cadena literal entre comillas*⟩⟨*separador*⟩]⟨*lista de:*⟨*variable*⟩

```
10 MODE 1
20 INPUT "Escriba los dos factores (separados por una coma): ",a,b
30 PRINT a;"por";b;"es";a*b
40 GOTO 20
run
```

Orden. Capta una entrada de datos por el canal especificado (por el **#0**, si no se especifica ninguno).

El primer signo de punto y coma es opcional e inhibe la función de retorno del carro/avance de línea que de otra forma se produciría al terminar de ejecutarse la instrucción.

El ⟨separador⟩ tiene que ser una coma o un punto y coma. El punto y coma hace que se escriba el signo de interrogación; la coma lo impide.

Si el dato introducido es de tipo incorrecto (por ejemplo, cuando se escribe la letra **O** en lugar del número **0** en la captación de un dato numérico), BASIC responde con el mensaje:

```
?Redo from start
```

y cualquier otro texto que se haya programado.

Toda introducción de datos a través del teclado debe terminar con la pulsación de la tecla INTRO.

Palabras clave asociadas: **INKEY$**, **LINE INPUT**.

INSTR

INSTR([⟨*posición de partida*⟩,]⟨*cadena en la que se busca*⟩,⟨*cadena buscada*⟩)

```
10 CLS:FOR n=1 TO 26
20 alfabeto$=alfabeto$+CHR$(n+64)
30 NEXT
40 INPUT "Escriba una letra: ",a$
50 b$=UPPER$(a$)
60 PRINT b$;" es la letra numero";
70 PRINT INSTR(alfabeto$,b$);
80 PRINT "del alfabeto":PRINT
90 GOTO 40
run
```

Función. Determina si la ⟨cadena buscada⟩ está contenida en la ⟨cadena en la que se busca⟩. Si lo está, da la posición del carácter de la ⟨cadena en la que se busca⟩ en el que se ha producido la primera coincidencia. Si no lo está, da el número **0**.

El parámetro opcional ⟨posición de partida⟩ indica en qué lugar de la ⟨ cadena en la que se busca⟩ debe empezar la búsqueda; debe ser una expresión entera que dé un valor comprendido entre 1 y 255.

Palabras clave asociadas: **LEFT$**, **MID$**, **RIGHT$**.

INT

INT(⟨*expresión numérica*⟩**)**

```
PRINT INT(-1.995)
-2
```

Función. Redondea el valor de la ⟨expresión numérica⟩ al número entero inferior, despreciando la parte decimal. Da el mismo valor que **FIX** cuando el argumento es positivo, pero una unidad menos que **FIX** cuando el argumento es negativo y no entero.

Palabras clave asociadas: **CINT**, **FIX**, **ROUND**.

JOY

JOY(⟨*expresión entera*⟩**)**

```
10 PRINT "Para detener el programa ";
20 PRINT "accione el joystick"
30 IF JOY(0)<> 0 THEN END
40 GOTO 10
run
```

Función. Lee bit a bit la situación del joystick especificado por la ⟨expresión entera⟩ (que debe valer **0** o **1**). Genera un valor decimal según la siguiente tabla:

	Bit	Decimal
0	Arriba	1
1	Abajo	2
2	Izquierda	4
3	Derecha	8
4	Disparo 2	16
5	Disparo 1	32

Así, en el programa ejemplo, si el botón principal de disparo (Disparo 2) del primer joystick está accionado mientras la empuñadura está inclinada hacia la izquierda, la función **JOY(0)** da el número decimal 20, que es la suma de 16 (Disparo 2) más 4 (Izquierda).

En el capítulo 'Para su referencia . . .' daremos más información acerca de los joysticks.

Palabras clave asociadas: **INKEY**.

KEY

KEY ⟨*código expansible*⟩,⟨*expresión literal*⟩

```
KEY 11,"border 13:paper 0:pen 1:ink 0,13:
ink 1,0:mode 2:list"+CHR$(13)
```

y ahora pulse la tecla [INTRO].

Orden. Asigna la ⟨expresión literal⟩ al ⟨código expansible⟩ especificado. El sistema maneja 32 códigos expansibles, del 0 al 31, identificados por los códigos de tecla 128 a 159. Los códigos de tecla del 128 (generado por la tecla **0** del teclado numérico) al 140 (generado por la tecla [INTRO] con [CTRL]) tienen asignadas implícitamente las funciones de escribir los dígitos del **0** al **9**, el punto, [INTRO] y **RUN"**[INTRO], respectivamente, pero se les pueden asignar otras ⟨expresiones literales⟩ diferentes con **KEY**. Los códigos expansibles 13 a 31 (códigos de tecla 141 a 159) tienen asignada implícitamente la cadena vacía, pero se los puede expandir con **KEY** para luego asignarlos a teclas concretas con **KEY DEF**.

El parámetro ⟨código expansible⟩ incluido en la orden **KEY** puede estar en el margen de 0 a 31, o bien en el margen 128 a 159 para reflejar directamente códigos de tecla. (Véase la ilustración del capítulo 'Para su referencia ...'.)

Las ⟨expresiones literales⟩ con las que se expandan los códigos expansibles pueden ocupar como máximo un total de 120 caracteres. Si se intenta sobrepasar ese límite se produce el error **'improper argument'** ('argumento inadecuado', error número 5).

Palabras clave asociadas: **KEY DEF**.

KEY DEF

KEY DEF ⟨*número de tecla*⟩,⟨*repetir*⟩[,⟨*normal*⟩[,⟨*mayúscula*⟩[,⟨*control*⟩]]]

```
KEY 159,"esta es la tecla TAB"
KEY DEF 68,1,159
```

Después de escribir estas órdenes, pulse la tecla [TAB].

Orden. Define los códigos de tecla que van a ser generados por la tecla de número especificado. El ⟨número de tecla⟩ tiene que estar en el margen de 0 a 79 (véanse los números de tecla en el diagrama grabado a la derecha del teclado, o bien el esquema del capítulo 'Para su referencia ...'). En la definición se pueden incluir los códigos de tecla que deben ser generados por la tecla cuando se la pulsa sola y cuando se la pulsa en combinación con [MAYS] y [CTRL]. Todos estos parámetros son opcionales.

El parámetro ⟨repetir⟩ puede tener los valores **1** (para que la tecla sea de repetición) o **0** (para que no lo sea). La velocidad de repetición se puede definir con **SPEED KEY**.

En el ejemplo anterior, primero se asigna una cadena literal al código **159** (equivalente al código expansible 31), y luego se especifica que la tecla número **68** (la tecla [TAB]) debe ser de repetición (**1**) y generar el código **159** cuando se la pulse sola.

La situación normal se restaura, en el ejemplo anterior, mediante la orden:

```
KEY DEF 68,0,9
```

ya que el 9 es el código ASCII normalmente generado por [TAB].

Palabras clave asociadas: **KEY**, **SPEED KEY**.

LEFT$

LEFT$(⟨*expresión literal*⟩,⟨*longitud deseada*⟩**)**

```
10 CLS
20 a$="AMSTRAD"
30 FOR n=1 TO 7
40 PRINT LEFT$(a$,n)
50 NEXT
run
```

Función. Produce una cadena que consta del número de caracteres dado por ⟨longitud deseada⟩ (margen de **0** a **255**) tomados de la izquierda de la ⟨expresión literal⟩. Si la longitud de la ⟨expresión literal⟩ es menor que la ⟨longitud deseada⟩, la cadena producida es la ⟨expresión literal⟩ completa.

Palabras clave asociadas: **MID$**, **RIGHT$**.

LEN

LEN(⟨*expresión literal*⟩**)**

```
10 LINE INPUT "Escriba una frase: ",a$
20 PRINT "La frase tiene";
30 PRINT LEN(a$);"caracteres."
run
```

Función. Da la longitud de la ⟨expresión literal⟩, es decir, el número de caracteres de que consta ésta.

LET

LET ⟨*variable*⟩=⟨*expresión*⟩

```
LET x=100
```

Orden. Asigna un valor a una variable. Es una reliquia de las primeras versiones de BASIC, innecesaria en el BASIC de AMSTRAD. En el ejemplo anterior habría bastado con escribir:

```
x=100
```

LINE INPUT

LINE INPUT [#⟨*expresión de número de canal*⟩,][;][⟨*cadena entre comillas*⟩⟨*separador*⟩] ⟨*variable literal*⟩

```
10 LINE INPUT "Escriba una linea de texto: ",a$
20 CLS
30 PRINT "La variable a$ es igual a: "
40 PRINT a$
run
```

Orden. Acepta la introducción de una línea entera de texto por el canal especificado (o por el **#0**, si no se especifica ninguno). El primer signo de punto y coma es opcional y, si se lo incluye, inhibe el retorno del carro/avance de línea que de otra forma se realizaría al terminar de ejecutarse la orden.

El ⟨separador⟩ tiene que ser una coma o un punto y coma. El punto y coma hace que se escriba el signo de interrogación; la coma lo impide.

La captación de datos a través del teclado debe terminar con la pulsación de la tecla [INTRO].

La captación de datos por el canal **#9** (cinta) termina cuando se recibe un carácter de retorno del carro o cuando se han recibido 255 caracteres (lo que antes ocurra).

Palabras clave asociadas: **INPUT**.

LIST

LIST [⟨*margen de números de línea*⟩][,#⟨*expresión de número de canal*⟩]

```
LIST 100-1000,#1
```

Orden. Envía el listado de las líneas especificadas al canal especificado. El canal implícito es el **#0**. La impresora es el canal **#8.** La acción de listar se puede suspender momentáneamente pulsando [ESC] una vez, tras lo cual se la puede reanudar pulsando la barra espaciadora o abandonar definitivamente pulsando [ESC] por segunda vez.

Si se omite el primer número de ⟨margen de números de línea⟩, se entiende 'desde el principio del programa hasta . . .'; si se omite el segundo número, se entiende '. . . hasta el final del programa'. Así,

```
LIST -200
```

lista las primeras líneas del programa, hasta la 200 inclusive. En cambio,

```
LIST 50-
```

lista desde la línea 50 en adelante. Finalmente,

```
LIST
```

lista el programa completo.

LOAD

LOAD ⟨*nombre de fichero*⟩[,⟨*expresión de dirección de memoria*⟩]

```
LOAD "fichero",&2AF8
```

Orden. Lee un programa de BASIC grabado en cinta y lo carga en la memoria, reemplazando al actual. Si se especifica la ⟨expresión de dirección de memoria⟩ opcional, **LOAD** carga un fichero binario en esa dirección, en lugar de hacerlo en la dirección a partir de la cual se lo grabó.

Si no se especifica nombre de fichero, BASIC intenta cargar el primer fichero válido que encuentra en la cinta. Si el primer carácter del nombre es !, BASIC suprime tanto este carácter como todos los mensajes que de otra forma generaría el sistema de cinta.

Los programas de BASIC protegidos no se pueden cargar por este método, pues el ordenador los borra inmediatamente de la memoria. Esos programas deben ser cargados con **RUN** o **CHAIN**.

Palabras clave asociadas: **CHAIN**, **CHAIN MERGE**, **MERGE**, **RUN**, **SAVE**.

LOCATE

LOCATE [#⟨*expresión de número de canal*⟩,]⟨*coordenada x*⟩,⟨*coordenada y*⟩

```
10 MODE 1
20 FOR n=1 TO 20
30 LOCATE n,n
40 PRINT CHR$(143);"posicion";
50 PRINT n;",";n
60 NEXT
run
```

Orden. Coloca el cursor de texto del canal indicado en la posición especificada por las coordenadas **x,y**. La posición **1,1** es el extremo superior izquierdo de la ventana. El canal **#0** es el canal implícito.

Palabras clave asociadas: **WINDOW**.

LOG

LOG(⟨*expresión numérica*⟩**)**

```
PRINT LOG(9999)
 9.21024037
```

Función. Calcula el logaritmo natural o neperiano de la ⟨expresión numérica⟩, la cual debe ser mayor que cero.

Palabras clave asociadas: **EXP**, **LOG10**.

LOG10

LOG10(⟨*expresión numérica*⟩**)**

```
PRINT LOG10(9999)
 3.99995657
```

Función. Calcula el logaritmo decimal de la ⟨expresión numérica⟩, la cual debe ser mayor que cero.

Palabras clave asociadas: **EXP**, **LOG**.

LOWER$

LOWER$(⟨*expresión literal*⟩**)**

```
10 a$="OBSERVE COMO SE CONVIERTEN LAS LETRAS A "
20 PRINT LOWER$(a$+"MINUSCULAS")
run
```

Función. Da una cadena literal que es copia de la ⟨expresión literal⟩ especificada, pero convirtiendo las letras mayúsculas en minúsculas y dejando inalterados los restantes caracteres. Esta función es útil en el proceso de datos que puedan contener minúsculas mezcladas con mayúsculas.

Palabras clave asociadas: **UPPER$**.

MAX

MAX(⟨*lista de:*⟨*expresión numérica*⟩**)**

```
10 n=66
20 PRINT MAX(1,n,3,6,4,3)
run
 66
```

Función. Da el valor máximo de los incluidos en la ⟨lista de:⟨expresión numérica⟩.

Palabras clave asociadas: **MIN**.

MEMORY

MEMORY ⟨*expresión de dirección de memoria*⟩

```
MEMORY &20AA
```

Orden. Determina el espacio de memoria utilizable por BASIC al fijar la dirección de su límite superior.

Palabras clave asociadas: **FRE, HIMEM, SYMBOL, SYMBOL AFTER.**

MERGE

MERGE ⟨*nombre del fichero*⟩

```
MERGE "nuevoprograma"
```

Orden. Lee un programa grabado en disco y lo superpone al actualmente almacenado en la memoria.

Las líneas del programa leído se superponen y sustituyen a las que hubiera en el antiguo con el mismo número.

Si no se especifica nombre de fichero, BASIC intenta cargar el primer fichero válido que encuentra en la cinta. Si el primer carácter del nombre es !, BASIC suprime tanto este carácter como todos los mensajes que de otra forma generaría el sistema de lectura de la cinta.

MERGE descarta los valores actuales de las variables y las funciones definidas por el usuario y abandona los ficheros que estén abiertos. Además, desactiva la intercepción de errores que pueda haber sido activada con **ON ERROR GOTO**, ejecuta implícitamente una orden **RESTORE**, reinicializa todas las características establecidas con **DEFINT**, **DEFREAL** y **DEFSTR** y abandona las instrucciones **FOR**, **WHILE** y **GOSUB** que estén activas.

Los ficheros protegidos (grabados con **,p**) no pueden ser mezclados por este procedimiento con el programa actual.

Palabras clave asociadas: **CHAIN, CHAIN MERGE, LOAD.**

MID$

MID$(⟨*expresión literal*⟩,⟨*posición de partida*⟩[,⟨*longitud de la subcadena*⟩]**)**

```
10 MODE 1:ZONE 3
20 a$="MASSACHUSETTS"
30 PRINT "Deletreame ";a$
40 PRINT "Vale ...":PRINT
50 FOR n=1 TO LEN(a$)
60 PRINT MID$(a$,n,1),
70 FOR t=1 TO 700:NEXT t,n
80 PRINT:PRINT
90 INPUT "Ahora escriba usted otra palabra: ",a$
100 GOTO 50
run
```

Función. Genera una cadena literal copiada de la ⟨expresión literal⟩. El número de caracteres copiados es el dado por ⟨longitud de la subcadena⟩; la copia empieza en el carácter especificado por ⟨posición de partida⟩. Si no se especifica la ⟨longitud de la subcadena⟩, se copian todos los caracteres restantes, desde ⟨posición de partida⟩ hacia la derecha.

Si la ⟨posición de partida⟩ es mayor que la longitud de la ⟨expresión literal⟩, se genera la cadena vacía. El margen de ⟨posición de partida⟩ es de 1 a 255. El de ⟨longitud de la subcadena⟩ es de 0 a 255.

Palabras clave asociadas: **LEFT$**, **RIGHT$**.

MID$

MID$(⟨*variable literal*⟩,⟨*posición de inserción*⟩[,⟨*longitud de cadena nueva*⟩]**)**
=⟨*expresión de cadena nueva*⟩

```
10 a$="Amstrad"
20 MID$(a$,3,2)="XX"
30 PRINT a$
run
AmXXrad
```

Orden. Reemplaza un número de caracteres de la ⟨variable literal⟩, dado por ⟨longitud de cadena nueva⟩, con caracteres tomados de ⟨expresión de cadena nueva⟩. La sustitución empieza en ⟨posición de inserción⟩.

Obsérvese que cuando se utiliza **MID$** como orden, su argumento debe ser una ⟨variable literal⟩; no vale, pues, una cadena literal constante tal como **"hola"**.

Palabras clave asociadas: **LEFT$**, **RIGHT$**.

MIN

MIN(⟨*lista de:*⟨*expresión numérica*⟩**)**

```
PRINT MIN(3,6,2.999,8,9)
 2.999
```

Función. Da el valor mínimo de los incluidos en la ⟨lista de:⟨expresión numérica⟩.

Palabras clave asociadas: **MAX**.

MOD

⟨*argumento*⟩ **MOD** ⟨*argumento*⟩

```
PRINT 10 MOD 3
 1
PRINT 10 MOD 5
 0
```

Operador. Da el resto de la división entera del primer ⟨argumento⟩ por el segundo.

MODE

MODE ⟨*expresión entera*⟩

```
10 m=m+1:IF m>2 THEN m=0
20 MODE m
30 PRINT"Este es el modo";m
40 PRINT"Pulse una tecla."
50 IF INKEY$="" THEN 50 ELSE 10
run
```

Orden. Cambia el modo de pantalla (0, 1 o 2) y la borra tiñéndola con la tinta número 0 (que puede no ser la tinta actualmente asignada al papel). Todas las pantallas de texto y gráficas se restauran (coincidentes con la pantalla completa); los cursores de texto y gráfico quedan en sus orígenes respectivos.

Palabras clave asociadas: **ORIGIN**, **WINDOW**.

MOVE

MOVE ⟨*coordenada x*⟩,⟨*coordenada y*⟩

```
10 MODE 1:TAG
20 x=RND*800-100:y=RND*430
30 MOVE x,y
40 PRINT "Estoy aqui";
50 GOTO 20
run
```

Orden. Lleva el cursor gráfico a la posición absoluta especificada por las coordenadas **x,y**.

Palabras clave asociadas: **DRAW, DRAWR, MOVER, ORIGIN, PLOT, PLOTR, TEST, TESTR, XPOS, YPOS.**

MOVER

MOVER ⟨*incremento de x*⟩,⟨*incremento de y*⟩

```
10 MODE 1:TAG:MOVE 0,16
20 PRINT "primero se sube";
30 FOR n=1 TO 8
40 MOVER -56,16
50 PRINT "sube";:NEXT:PRINT " y luego se baja";
60 FOR n=1 TO 8
70 MOVER -56,-16
80 PRINT "baja";:NEXT
run
```

Orden. Lleva el cursor gráfico a un punto cuya posición, referida a la actual, viene dada por ⟨incremento de x⟩e ⟨incremento de y⟩.

Palabras clave asociadas: **DRAW, DRAWR, MOVE, ORIGIN, PLOT, PLOTR, TEST, TESTR, XPOS, YPOS.**

NEW

NEW

```
NEW
```

Orden. Borra el programa de BASIC actualmente residente en la memoria. No se pierden las definiciones de teclas ni se modifican las características de la pantalla (**MODE, PEN, PAPER, INK**, etc.) ni se borra la pantalla.

NEXT

NEXT [⟨*lista de:*⟨*variable*⟩]

```
10 FOR a=1 TO 3
20 FOR b=0 TO 26
30 MODE 1
40 PEN a:BORDER b
50 PRINT "pen";a;" border";b
60 FOR c=1 TO 500
70 NEXT c,b,a
run
```

Orden. Señala el final del bucle **FOR** ... **NEXT**. La orden **NEXT** puede ser anónima o mencionar el nombre de la variable del **FOR** correspondiente. Obsérvese en el ejemplo anterior que las variables de la lista deben estar en orden inverso al de sus correspondientes órdenes **FOR**, de modo que los bucles no queden solapados, sino anidados.

Palabras clave asociadas: **FOR**, **STEP**, **TO**.

NOT

NOT ⟨*argumento*⟩

```
IF NOT "juan"<"pepe" THEN PRINT "verdadero" ELSE PRINT "falso"
falso
IF NOT "gato">"perro" THEN PRINT "verdadero" ELSE PRINT "falso"
verdadero
...
PRINT NOT -1
 0
PRINT NOT 0
-1
```

Operador. Realiza la negación lógica sobre enteros inviertiendo todos los bits del argumento.

En la parte 2 del capítulo 'Cuando usted guste ...' daremos más amplia información sobre las operaciones lógicas.

Palabras clave asociadas: **AND**, **OR**, **XOR**.

ON BREAK GOSUB

ON BREAK GOSUB ⟨*número de línea*⟩

```
10 ON BREAK GOSUB 40
20 PRINT "Programa funcionando"
30 GOTO 20
40 CLS:PRINT "Pulsando dos veces [ESC] ";
50 PRINT"se invoca la subrutina"
60 FOR t=1 TO 2000:NEXT
70 RETURN
run
```

Orden. Provoca el salto a la subrutina especificada por ⟨número de línea⟩ cuando se pulsa dos veces la tecla [ESC].

Palabras clave asociadas: **ON BREAK STOP**, **RETURN**.

ON BREAK STOP

ON BREAK STOP ⟨*número de línea*⟩

```
10 ON BREAK GOSUB 40
20 PRINT "Programa funcionando"
30 GOTO 20
40 CLS:PRINT "Pulsando dos veces [ESC] ";
50 PRINT"se invoca la subrutina"
60 FOR t=1 TO 2000:NEXT
65 ON BREAK STOP
70 RETURN
run
```

Orden. Anula la orden **ON BREAK GOSUB**, permitiendo así que el programa pueda ser interrumpido normalmente con la tecla [ESC]. En el programa ejemplo, la orden **ON BREAK GOSUB** sólo actúa una vez, pues la línea 65 de la rutina de interrupción la anula.

Palabras clave asociadas: **ON BREAK GOSUB**.

ON ERROR GOTO

ON ERROR GOTO ⟨*número de línea*⟩

```
10 ON ERROR GOTO 60
20 CLS:PRINT "Si detecto un error, ";
30 PRINT "listare el programa"
40 FOR t=1 TO 4000:NEXT
50 GOTO 100
60 PRINT "Hay un error en la linea";
70 PRINT ERL:PRINT:LIST
run
```

Orden. Provoca el salto a la línea especificada cuando se detecta un error en el programa.

La forma **ON ERROR GOTO 0** desactiva la gestión de errores programada por el usuario y activa la normal de BASIC.

Véase también la orden **RESUME**.

Palabras clave asociadas: **ERL, ERR, ERROR, RESUME.**

ON ⟨expresión⟩ GOSUB

ON ⟨*selector*⟩ **GOSUB** ⟨*lista de:*⟨*número de línea*⟩

```
10 PAPER 0:PEN 1:INK 0,1
20 CLS:PRINT "MENU DE OPCIONES":PRINT
30 PRINT "1. Cambiar borde":PRINT
40 PRINT "2. Cambiar pluma":PRINT
50 PRINT "3. Cambiar modo":PRINT
60 INPUT "Elija del 1 al 3: ",x
70 ON x GOSUB 90,110,130
80 GOTO 20
90 b=b-1:IF b<0 THEN b=26
100 BORDER b:RETURN
110 p=p-1:IF p<2 THEN p=26
120 INK 1,p:RETURN
130 m=m-1:IF m<0 THEN m=2
140 MODE m:RETURN
run
```

Orden. Determina, en función del valor de ⟨selector⟩, qué subrutina de BASIC debe ser invocada. El margen de valores de ⟨selector⟩ es de 0 a 255. La rutina elegida es la que se encuentra en la lista en el lugar indicado por ⟨selector⟩. Así, en el ejemplo anterior, si x=1, se invoca la rutina de la línea 90; si x=2, la línea 110; y si x=3, la de la línea 130.

Si el valor de ⟨selector⟩ es cero o mayor que el número de líneas citadas en la ⟨lista de números de línea⟩, no se invoca ninguna subrutina.

Palabras clave asociadas: **RETURN**.

ON ⟨expresión⟩ GOTO

ON ⟨*selector*⟩ **GOTO** ⟨*lista de:⟨número de línea*⟩

```
10 CLS:PRINT "MENU DE OPCIONES":PRINT
20 PRINT "1. Listar programa":PRINT
30 PRINT "2. Ampliar programa":PRINT
40 PRINT "3. Catalogo de la cinta":PRINT
50 INPUT "Elija del 1 al 3: ",x
60 ON x GOTO 80,90,100
70 GOTO 10
80 LIST
90 AUTO 110
100 CAT
run
```

Orden. Determina, en función del valor de ⟨selector⟩, a qué línea del programa debe saltar la ejecución. El margen de valores de ⟨selector⟩ es de 0 a 255. La línea elegida es la que se encuentra en la lista en el lugar indicado por ⟨selector⟩. Así, en el ejemplo anterior, si x=1 se salta a la línea 80; si x=2, a la línea 90; y si x=3, a la línea 100.

Si el valor de ⟨selector⟩ es cero o mayor que el número de líneas citadas en la ⟨lista de números de línea⟩, no se produce el salto.

Palabras clave asociadas: **RETURN**.

ON SQ GOSUB

ON SQ(⟨*canal*⟩**) GOSUB** ⟨*número de línea*⟩

```
10 ENV 1,15,-1,1
20 ON SQ(1) GOSUB 70
30 MODE 0
40 FOR x=1 TO 13:CALL &BD19
50 PLOT 0,0,x:DRAW 640,400:PLOT 640,0:DRAW 0,400
60 NEXT:GOTO 40
70 READ s:IF s=0 THEN RESTORE:GOTO 70
80 SOUND 1,s,25,15,1
90 ON SQ(1) GOSUB 70:RETURN
100 DATA 50,60,90,100,35,200,24,500,0
run
```

Orden. Invoca la subrutina especificada cuando hay espacio libre en la cola de sonidos. El parámetro ⟨canal⟩ deber ser una expresión entera que dé uno de los siguientes valores:

1 para el canal A
2 para el canal B
4 para el canal C

En la parte 2 del capítulo 'Cuando usted guste ...' daremos más información sobre los sonidos.

Palabras clave asociadas: **RETURN, SOUND, SQ.**

OPENIN

OPENIN ⟨*nombre de fichero*⟩

```
10 REM abrir un fichero de cinta para entrada
20 OPENIN "!datos":INPUT #9,a,a$
30 CLOSEIN:PRINT "Los dos valores son:"
40 PRINT:PRINT a,a$
run
```

Orden. Abre, en dirección de entrada, un fichero grabado en cinta para hacerlo accesible al programa. El fichero tiene que ser de tipo ASCII.

Si el primer carácter del nombre del fichero es !, BASIC suprime tanto este carácter como todos los mensajes que de otra forma generaría el sistema de cinta.

El programa anterior sólo funciona si se ha creado previamente el fichero (por ejemplo, con el programa siguiente).

Palabras clave asociadas: **CLOSEIN, EOF.**

OPENOUT

OPENOUT ⟨*nombre de fichero*⟩

```
10 REM abrir un fichero en la cinta para salida
20 INPUT "Escriba un numero: ",a
30 INPUT "Escriba una palabra: ",a$
40 OPENOUT "!datos"
50 WRITE #9,a,a$
60 CLOSEOUT:PRINT "Datos grabados en cinta"
run
```

Orden. Abre, en dirección de salida, un fichero para su grabación en cinta.

Si el primer carácter del nombre del fichero es !, BASIC suprime tanto este carácter como todos los mensajes que de otra forma generaría el sistema de cinta.

Palabras clave asociadas: **CLOSEOUT.**

OR

⟨*argumento*⟩ **OR** ⟨*argumento*⟩

```
IF "juan"<"pepe" OR "perro">"gato" THEN PRINT "verdadero" ELSE PRINT "falso"
verdadero
IF "pepe"<"juan" OR "gato">"perro" THEN PRINT "verdadero" ELSE PRINT "falso"
falso
IF "juan"<"pepe" OR "gato">"perro" THEN PRINT "verdadero" ELSE PRINT "falso"
verdadero
...
PRINT 1 OR 1
 1
PRINT 0 OR 0
 0
PRINT 1 OR 0
 1
```

Operador. Realiza la operación lógica 'o' bit a bit entre enteros. El resultado es 1 a menos que ambos bits sean 0.

Para más amplia información sobre operaciones lógicas, consulte la parte 2 del capítulo titulado 'Cuando usted guste ...'.

Palabras clave asociadas: **AND**, **NOT**, **XOR**.

ORIGIN

ORIGIN ⟨*x*⟩,⟨*y*⟩[,⟨*izquierda*⟩,⟨*derecha*⟩,⟨*arriba*⟩,⟨*abajo*⟩]

```
10 CLS:BORDER 13
20 ORIGIN 0,0,50,590,350,50
30 DRAW 540,350
40 GOTO 20
```

Orden. Traslada el origen de gráficos (nuevo punto **0,0**) a la posición especificada por ⟨x⟩ e ⟨y⟩.

Al mismo tiempo se pueden especificar las dimensiones de una ventana gráfica dando los cuatro últimos parámetros (opcionales). Si estos parámetros definen posiciones externas a la pantalla, el borde de la ventana será el borde de la pantalla.

Palabras clave asociadas: **CLG**, **WINDOW**.

OUT

OUT ⟨*número de puerta*⟩,⟨*expresión entera*⟩

```
OUT &F8F4,&FF
```

Orden. Envía el valor de ⟨expresión entera⟩ (margen de 0 a 255) a la dirección especificada por ⟨número de puerta⟩.

Esta orden no debe ser utilizada si no se sabe cuáles son sus efectos.

Palabras clave asociadas: **INP**, **WAIT**.

PAPER

PAPER [#⟨*expresión de número de canal*⟩,]⟨*tinta*⟩

```
10 MODE 0:PEN 0:INK 0,13
20 FOR p=1 TO 15
30 PAPER p:CLS
40 LOCATE 7,12:PRINT "paper";p
50 FOR t=1 TO 500:NEXT t,p
run
```

Orden. Establece qué tinta se va a utilizar para escribir el fondo (papel) de los caracteres. Si los caracteres se envían a la pantalla de texto, la celda se rellena con el color actual del papel antes de escribir el carácter (a menos que se haya elegido el modo transparente).

Si se omite la ⟨expresión de número de canal⟩ se supone el **#0**.

El número de tintas utilizables simultáneamente depende del modo de pantalla.

Palabras clave asociadas: **INK**, **PEN**, **WINDOW**.

PEEK

PEEK(⟨*expresión de dirección de memoria*⟩**)**

```
10 MODE 1:ZONE 7
20 WINDOW 1,40,1,3:WINDOW #1,1,40,4,25
30 LOCATE 13,1:PRINT"MEMORIA"
40 PRINT:PRINT"Direccion"
50 LOCATE 20,3:PRINT"Contenido"
60 FOR n=0 TO 65535
70 p=PEEK(n)
80 PRINT #1,n,"(&";HEX$(n);")";
90 PRINT #1,TAB(20);p,"(&";HEX$(p);")"
100 NEXT
run
```

Función. Da el contenido de la posición de memoria especificada por el argumento, el cual debe estar en el margen de &0000 a &FFFF (0 a 65535). En todos los casos, el valor producido será el contenido de la posición de memoria de la RAM (no de la ROM) y estará entre &00 y &FF (entre 0 y 255).

Palabras clave asociadas: **POKE**.

PEN

PEN [#⟨*expresión de número de canal*⟩,]⟨*tinta*⟩

```
10 MODE 0:PAPER 0:INK 0,13
20 FOR p=1 TO 15
30 PEN p:PRINT SPACE$(47);"pen";p
40 FOR t=1 TO 500:NEXT t,p:GOTO 20
run
```

Orden. Establece la ⟨tinta⟩ (margen de **0** a **15**) con la que se escribirán los caracteres en lo sucesivo en el canal especificado (en el **#0** si no se especifica ninguno).

Palabras clave asociadas: **INK**, **PAPER**.

PI

PI

```
PRINT PI
 3.14159265
```

Función. Da el valor del número π.

Palabras clave asociadas: **DEG**, **RAD**.

PLOT

PLOT ⟨*coordenada x*⟩,⟨*coordenada y*⟩[,⟨*tinta*⟩]

```
10 MODE 1:BORDER 0:PAPER 0:PEN 1
20 INK 0,0:INK 1,26:INK 2,13,26:DEG
30 FOR x=1 TO 360:ORIGIN 320,200
40 DRAW 50*COS(x),50*SIN(x),1
50 PLOT 100*COS(x),25*SIN(x):NEXT
60 ORIGIN 0,0:t=TIME+700:WHILE TIME<t
70 PLOT RND*640,RND*400:WEND
80 PLOT RND*640,RND*400,2
90 GOTO 90
run
```

Orden. Dibuja un punto en la pantalla gráfica, en la posición absoluta especificada por los parámetros ⟨coordenada x⟩ y ⟨coordenada y⟩. También se puede especificar la ⟨tinta⟩ con la que se debe dibujar (margen de **0** a **15**).

Palabras clave asociadas: **DRAW**, **DRAWR**, **MOVE**, **MOVER**, **ORIGIN**, **PLOTR**, **TEST**, **TESTR**, **XPOS**, **YPOS**.

PLOTR

PLOTR ⟨*incremento de x*⟩,⟨*incremento de y*⟩[,⟨*tinta*⟩]

```
10 REM dibujo de rectas con las teclas del cursor
20 BORDER 0
30 MODE 1:PLOT 320,200,1
40 IF INKEY(0)=0 THEN PLOTR 0,1
50 IF INKEY(1)=0 THEN PLOTR 1,0
60 IF INKEY(2)=0 THEN PLOTR 0,-1
70 IF INKEY(8)=0 THEN PLOTR -1,0
80 IF INKEY(9)=0 THEN 30:REM copia = borrar
90 GOTO 40
run
```

Orden. Dibuja un punto en la pantalla gráfica en una posición que, referida a la actual, viene dada por los parámetros ⟨incremento de x⟩ e ⟨incremento de y⟩. Se puede utilizar el parámetro opcional ⟨tinta⟩ para cambiar la tinta de la pluma gráfica (margen de **0** a **15**).

Palabras clave asociadas: **DRAW**, **DRAWR**, **MOVE**, **MOVER**, **ORIGIN**, **PLOT**, **TEST**, **TESTR**, **XPOS**, **YPOS**.

POKE

POKE ⟨*expresión de dirección de memoria*⟩,⟨*expresión entera*⟩

```
10 FOR m=49152 TO 65535
20 POKE m,100
30 NEXT
run
```

Orden. Escribe el valor de la ⟨expresión entera⟩ (margen de 0 a 255) en la memoria de la máquina (RAM), en la posición especificada por el parámetro ⟨expresión de dirección de memoria⟩.

Esta orden no se debe utilizar si no se sabe cuáles son sus efectos.

Palabras clave asociadas: **PEEK**.

POS

POS(#⟨*expresión de número de canal*⟩**)**

```
PRINT POS(#0)
 1
```

Función. Da la posición horizontal del cursor de texto referida al borde izquierdo de la ventana. El parámetro ⟨expresión de número de canal⟩ tiene que ser especificado, aunque sea el **#0**.

POS(#8) da la posición horizontal de la cabeza impresora, siendo 1 la posición del extremo izquierdo.

POS(#9) da la posición lógica en el canal de la cinta, esto es, el número de caracteres tipográficos enviados a la cinta después del último retorno del carro.

Palabras clave asociadas: **VPOS**, **WINDOW**.

PRINT

PRINT [#⟨*expresión de número de canal*⟩,][⟨*lista de:*⟨*elemento que se escribe*⟩]

```
10 a$="corta"
20 b$="esta cadena es mas larga"
30 PRINT a$;a$
40 PRINT a$,a$
50 PRINT
60 PRINT b$;b$
70 PRINT b$,b$
run
```

Orden. Envía los ⟨elementos que se escriben⟩ al canal especificado (o al **#0**, si no se especifica ninguno).

Cuando se utiliza un signo de punto y coma (;) para separar los elementos de la lista, el ordenador escribe cada elemento inmediatamente después del anterior; no obstante, BASIC comprueba si el elemento siguiente cabe en la misma línea y, si no cabe, salta a la línea siguiente a pesar del signo de punto y coma.

Análogamente, la coma (,) especifica que el siguiente elemento se debe escribir en la siguiente zona de escritura; no obstante, BASIC comprueba si con el elemento actual se ha rebasado la anchura de la zona de escritura y, si es así, escribe el siguiente elemento en una zona más a la derecha.

PRINT SPC
PRINT TAB

PRINT [#⟨*expresión de número de canal*⟩,][⟨*lista de:*⟨*elemento que se escribe*⟩][;]
[**SPC(**⟨*expresión entera*⟩**)**][⟨*lista de:*⟨*elemento que se escribe*⟩]

PRINT [#⟨*expresión de número de canal*⟩,][⟨*lista de:*⟨*elemento que se escribe*⟩][;]
[**TAB(**⟨*expresión entera*⟩**)**][⟨*lista de:*⟨*elemento que se escribe*⟩]

```
10 PRINT "Esta es la funcion SPC"
20 FOR x=6 TO 15
30 PRINT SPC(5)"a";SPC(x)"b"
40 NEXT
50 PRINT "Esta es la funcion TAB"
60 FOR x=6 TO 15
70 PRINT TAB(5)"a";TAB(x)"b"
80 NEXT
run
```

SPC escribe el número de espacios especificado por la ⟨expresión entera⟩ y luego, inmediatamente a su derecha, el siguiente ⟨elemento⟩ (en el supuesto de que quepa en la misma línea). Por consiguiente, no es necesario poner el signo de punto y coma después de **SPC**.

TAB avanza (escribiendo espacios) hasta la posición especificada por la ⟨expresión entera⟩, referida al borde izquierdo de la ventana, y luego, inmediatamente a la derecha de esa posición, escribe el siguiente ⟨elemento⟩ (en el supuesto de que quepa en la misma línea). Por consiguiente, no es necesario poner el signo de punto y coma después de **TAB**. Si la posición actual es mayor que la especificada, primero se realiza un retorno de carro y avance de línea y después el avance de los espacios necesarios.

PRINT USING

PRINT [#⟨*expresión de número de canal*⟩][⟨*lista de:*⟨*elemento que se escribe*⟩][;]
[**USING** ⟨*plantilla de formato*⟩][⟨*separador*⟩⟨*expresión*⟩

```
10 FOR x=1 TO 10
20 n=100000*(RND↑5)
30 PRINT "Precio";USING "########,.##";n
40 NEXT
run
```

PRINT USING permite especificar el formato de la expresión escrita con **PRINT**. Esto se consigue especificando una ⟨plantilla de formato⟩ a la que la escritura ha de amoldarse. El ⟨separador⟩ es una coma o un punto y coma. La ⟨plantilla de formato⟩ es una cadena literal construida con los siguientes 'especificadores de formato':

Formatos numéricos

Para el número en sí:

Cada # especifica una posición (para un dígito).
Ejemplo: ######

. Especifica la posición del punto decimal.
Ejemplo: ######.##

, (Especifica una posición.) Sólo puede aparecer antes del punto decimal. Especifica que los dígitos que queden a la izquierda del punto decimal deben separarse mediante comas en grupos de tres.
Ejemplo: #####,.##

Para antes y después del número:

$$ (Especifica dos posiciones.) Especifica que se debe escribir un signo $ inmediatamente antes del primer dígito o del punto decimal, después del signo + o – (si lo hay). Obsérvese que el signo $ ocupará una de las posiciones para dígitos.
Ejemplo: $$######,.##

**$ (Especifica tres posiciones.) Actúa como las opciones ** y $$ combinadas, es decir, con asteriscos por la izquierda y el signo $.
Ejemplo: **$######,.##

+ Especifica que se debe escribir el signo + o el – (según corresponda). Si se incluye el signo + al principio de la plantilla, el signo se escribirá inmediatamente antes del número (y del signo $, si se lo ha especificado). Si se incluye el signo + al final de la plantilla, el signo se escribe a la derecha del número (y del exponente, si lo hay).
Ejemplo: +####.####

– Este signo sólo se puede poner al final de la plantilla. Especifica que, si el número es negativo, se debe escribir el signo – a su derecha (y a la derecha del exponente, si lo hay). Si el número es positivo, a su derecha se escribe un espacio. Si no se especifica esto, se escribe automáticamente un signo – a la izquierda de todos los números negativos.
Ejemplo: ####.####–

↑↑↑↑ Especifica que el número se debe escribir en forma exponencial. Los signos ↑↑↑↑ se deben poner a la derecha de las posiciones para dígitos, pero a la izquierda de los signos + y – (si los hay).
Ejemplo: ##.####↑↑↑↑+

La ⟨plantilla de formato⟩ para números no puede contener más de 20 caracteres. Los números son redondeados antes de escribirlos con los dígitos especificados.

Si el formato es demasiado pequeño para el número que se ha de escribir, por ejemplo

PRINT USING "####";12345678

el número no se trunca, sino que se lo escribe entero, precedido de un signo % para indicar el 'fallo de formato'.

Formatos literales

```
10 CLS:a$="abcdefghijklmnopq"
20 PRINT "expresion de entrada = ";a$
30 PRINT:PRINT "con especificador ! = ";
40 PRINT USING "!";a$
50 PRINT:PRINT "con especificador \espacios\ = ";
60 PRINT USING "\          \";a$
70 PRINT:PRINT "con especificador & = ";
80 PRINT USING "&";a$
90 GOTO 90
run
```

! Especifica que solamente se debe escribir el primer carácter de la cadena.
Ejemplo: !

\⟨espacios⟩\
Especifica que solamente se deben escribir los x primeros caracteres de la cadena, siendo x la longitud de la plantilla (incluidas las barras inclinadas hacia la izquierda).
Ejemplo: \ \

& Especifica que se debe escribir la cadena completa sin modificarla.
Ejemplo: &

La ⟨plantilla de formato⟩ para cadenas literales no puede contener más de 255 caracteres.

Las plantillas de formato, tanto si son numéricas como si son literales, pueden ser representadas por variables literales; por ejemplo:

```
10 a$="$$######,.##"
20 b$="!"
30 PRINT USING a$;12345.6789
40 PRINT USING b$;"kilobytes"
run
```

En la parte 2 del capítulo 'Cuando usted guste ...' daremos más información acerca de los formatos de impresión.

Palabras clave relacionadas: **SPC, TAB, USING, ZONE.**

RAD

RAD

```
RAD
```

Orden. Hace que BASIC entienda los argumentos de las funciones trigonométricas en radianes. Ésta es la situación implícita que se establece en el momento de encender la máquina, cuando se la reinicializa y cuando se ejecutan las órdenes **NEW**, **CLEAR**, **LOAD**, **RUN**, etc.

Palabras clave asociadas: **ATN**, **COS**, **DEG**, **SIN**, **TAN**.

RANDOMIZE

RANDOMIZE [⟨*expresión numérica*⟩]

```
RANDOMIZE 123.456
PRINT RND
 0.258852139
```

Orden. El generador de números aleatorios de BASIC produce una sucesión pseudo-aleatoria en la que cada número depende del anterior. La sucesión es siempre la misma. **RANDOMIZE** hace que los números aleatorios se tomen de esa sucesión a partir de la posición dada por la ⟨expresión numérica⟩ o, si se omite ésta, por un valor que el usuario debe teclear.

RANDOMIZE TIME produce una sucesión que es muy difícil de repetir.

Palabras clave asociadas: **RND**.

READ

READ ⟨*lista de:*⟨*variable*⟩

```
10 FOR n=1 TO 8
20 READ a$,c
30 PRINT a$;" ";:SOUND 1,c:NEXT
40 DATA en,478,todas,426,las,379,escalas,358
50 DATA musicales,319,hay,284,8,253,notas,239
run
```

Orden. Lee datos de las listas precedidas de **DATA** y los asigna a variables, actualizando automáticamente el 'puntero' de forma que éste señale el siguiente dato. Se puede utilizar la orden **RESTORE** para hacer que el puntero señale el primer dato de una determinada lista **DATA**.

En la parte 2 del capítulo titulado 'Cuando usted guste . . .' daremos más amplia información a este respecto.

Palabras clave asociadas: **DATA**, **RESTORE**.

RELEASE

RELEASE ⟨*canales de sonido*⟩

```
10 SOUND 65,1000,100
20 PRINT "Pulse R para liberar el sonido"
30 IF INKEY(50)=-1 THEN 30
40 RELEASE 1
run
```

Orden. Libera los canales de sonido que han sido retenidos en una orden **SOUND**.

El parámetro ⟨canales de sonido⟩ tiene que dar un valor entero comprendido entre 1 y 7. Su significado es el siguiente:

1 libera el canal A
2 libera el canal B
3 libera los canales A y B
4 libera el canal C
5 libera los canales A y C
6 libera los canales B y C
7 libera los canales A, B y C

En la parte 2 del capítulo 'Cuando usted guste . . .' daremos más amplia información acerca de los sonidos.

Palabras clave asociadas: **SOUND**.

REM

REM ⟨*resto de la línea*⟩

```
10 REM Caza a muerte de los invasores del
   hiperespacio intergalactico, por AMSOFT
20 REM Copyright AMSOFT 1985
```

Orden. Precede a las observaciones o anotaciones al programa. Hace que BASIC ignore el resto de la línea, en el cual pueden figurar caracteres cualesquiera, incluso el signo de dos puntos (:), que normalmente se utiliza para separar instrucciones.

En lugar de **:REM** se puede poner una comilla sencilla (') en todos los casos salvo en las líneas de **DATA**, en las que el signo ' se interpreta como integrante de una cadena literal.

REMAIN

REMAIN (⟨*número de temporizador*⟩)

```
10 AFTER 500,1 GOSUB 40
20 AFTER 100,2 GOSUB 50
30 PRINT "Programa funcionando":GOTO 30
40 REM esta subrutina no sera invocada porque
   lo impide la linea 80
50 PRINT:PRINT "El temporizador 1 va a ser ";
60 PRINT "inhibido por REMAIN."
70 PRINT "Unidades de tiempo que quedaban:";
80 PRINT REMAIN(1)
run
```

Función. Da la cuenta restante en el temporizador especificado (margen de **0** a **3**) y lo desactiva.

En la parte 2 del capítulo 'Cuando usted guste ...' daremos más información acerca de las interrupciones.

Palabras clave asociadas: **AFTER, DI, EI, EVERY**.

RENUM

RENUM [⟨*nuevo número de línea*⟩][,[⟨*antiguo número de línea*⟩][,⟨*incremento*⟩]]

```
10 CLS
20 REM esta linea sera la 123
30 REM esta linea sera la 124
40 REM esta linea sera la 125
RENUM 123,20,1
LIST
```

Orden. Renumera las líneas del programa.

El parámetro ⟨antiguo número de línea⟩ especifica el número de línea actual en el cual debe comenzar la renumeración. Si se omite ese parámetro, la renumeración se realiza desde el principio del programa.

El parámetro ⟨nuevo número de línea⟩ especifica qué número se va a dar a la primera línea renumerada. Si se omite este parámetro, el programa renumerado empieza en la línea 10.

El parámetro ⟨incremento⟩ especifica el salto que habrá entre cada dos números de línea sucesivos. Si se lo omite, el incremento es 10.

RENUM modifica correctamente los números de línea citados en las instrucciones **GOTO**, **GOSUB** y similares. Sin embargo, no se modifican las referencias a números de línea que estén contenidas en expresiones literales, tales como las que intervienen en las órdenes **KEY**, ni tampoco las contenidas en líneas **REM**, ni las ⟨expresiones de número de línea⟩ incluidas en **CHAIN** y **CHAIN MERGE**.

Los números de línea válidos son los comprendidos entre **1** y **65535**.

Palabras clave asociadas: **DELETE**, **LIST**.

RESTORE

RESTORE [⟨*número de línea*⟩]

```
10 READ s$:PRINT a$;" ";
20 RESTORE 50
30 FOR t=1 TO 500:NEXT:GOTO 10
40 DATA restaure los datos para leerlos una vez
50 DATA y otra
run
```

Orden. Hace que el 'puntero' de datos 'apunte' hacia el principio de la línea de **DATA** especificada por el parámetro opcional ⟨número de línea⟩. Si se omite el parámetro, el puntero señala la primera línea de **DATA** del programa.

En la parte 2 del capítulo 'Cuando usted guste ...' daremos más información a este respecto.

Palabras clave asociadas: **DATA**, **READ**.

RESUME

RESUME [⟨*número de línea*⟩]

```
10 ON ERROR GOTO 60
20 FOR x=10 TO 0 STEP -1:PRINT 1/x:NEXT
30 END
40 PRINT "llegamos hasta aqui despues del error"
50 END
60 PRINT "error numero";ERR;"en la linea";ERL
70 RESUME 40
run
```

Orden. Reanuda la ejecución normal del programa una vez detectado un error y procesado por una orden **ON ERROR GOTO**. Si se omite el ⟨número de línea⟩, el programa continúa en la misma línea en la que se produjo el error. Suprima el ⟨número de línea⟩ en el ejemplo anterior y ejecute nuevamente el programa:

```
70 RESUME
run
```

Palabras clave asociadas: **ERL, ERR, ERROR, ON ERROR GOTO, RESUME NEXT.**

RESUME NEXT

RESUME NEXT

```
10 ON ERROR GOTO 90
20 PRINT "Pulse siempre INTRO"
30 INPUT "1";a
40 INPUT "2";a
50 inpot "3";a REM error de sintaxis
60 INPUT "4";a
70 INPUT "5";a
80 END
90 PRINT "error numero";ERR;"en la linea";ERL
100 RESUME NEXT
run
```

Orden. Reanuda la ejecución normal del programa una vez detectado un error y procesado por una orden **ON ERROR GOTO.**

RESUME NEXT reanuda la ejecución en la línea siguiente a aquélla en la que se detectó el error.

Palabras clave asociadas: **ERL, ERR, ERROR, ON ERROR GOTO, RESUME.**

RETURN

RETURN

```
10 GOSUB 50:PRINT "despues del GOSUB":END
50 FOR n=1 TO 20
60 PRINT "subrutina"
70 NEXT:PRINT
80 RETURN
run
```

Orden. Señala el final de una subrutina. BASIC retorna de la subrutina a la instrucción inmediatamente posterior a aquélla que invocó la subrutina.

Palabras clave asociadas: **GOSUB**.

RIGHT$

RIGHT$(⟨*expresión literal*⟩,⟨*longitud deseada*⟩**)**

```
10 MODE 0:a$="Ordenador CPC464"
20 FOR n=1 TO 16:LOCATE 21-n,n
30 PRINT RIGHT$(a$,n)
40 NEXT
run
```

Función. Produce una cadena que consta del número de caracteres dado por ⟨longitud deseada⟩ (margen de 0 a 255) tomados de la derecha de la ⟨expresión literal⟩. Si la longitud de la ⟨expresión literal⟩ es menor que la ⟨longitud deseada⟩, la cadena producida es la ⟨expresión literal⟩ completa.

Palabras clave asociadas: **LEFT$**, **MID$**.

RND

RND[(⟨*expresión numérica*⟩**)]**

```
10 RANDOMIZE
20 FOR x=1 TO -1 STEP -1
30 PRINT "parametro de rnd =";x
40 FOR n=1 TO 6
50 PRINT RND(X)
60 NEXT n,x
run
```

Función. Da el siguiente número aleatorio de la sucesión en el caso de que la ⟨expresión numérica⟩ tenga valor positivo o no haya sido especificada.

Si la ⟨expresión numérica⟩ toma valor cero, **RND** vuelve a dar el anterior número aleatorio generado.

Si la ⟨expresión numérica⟩ toma un valor negativo, se inicia una nueva sucesión aleatoria y **RND** da su primer número.

Palabras clave asociadas: **RANDOMIZE**.

ROUND

ROUND(⟨*expresión numérica*⟩[,⟨*decimales*⟩]**)**

```
10 FOR n=4 TO -4 STEP -1
20 PRINT ROUND(1234.5678,n),
30 PRINT "con ⟨decimales⟩ =";n
40 NEXT
run
```

Función. Redondea la ⟨expresión numérica⟩ a un número de decimales o potencia de diez especificada por ⟨decimales⟩. Si ⟨decimales⟩ es menor que cero, se redondea la ⟨expresión numérica⟩ para dar un número entero seguido de tantos ceros antes del punto decimal como indique el parámetro ⟨decimales⟩.

Palabras clave asociadas: **ABS**, **CINT**, **FIX**, **INT**.

RUN

RUN ⟨*expresión literal*⟩

```
RUN "miprograma"
```

Orden. Carga un programa de BASIC o un programa binario que esté grabado en cinta e inicia su ejecución. Borra el programa anteriormente residente en la memoria.

Si no se especifica nombre de fichero, BASIC intenta cargar el primer fichero válido que encuentra en la cinta. Si el primer carácter del nombre es !, BASIC suprime tanto este carácter como todos los mensajes que de otra forma generaría el sistema de cinta.

Los programas de BASIC protegidos pueden ser ejecutados por este procedimiento.

Palabras clave asociadas: **LOAD**.

RUN

RUN [⟨*número de línea*⟩]

```
RUN 200
```

Orden. Inicia la ejecución del programa de BASIC actual a partir de la línea especificada por ⟨número de línea⟩, o a partir de la primera línea del programa si se omite el parámetro.

Los programas protegidos no pueden ser ejecutados por este procedimiento.

Palabras clave asociadas: **CONT**, **END**, **STOP**.

SAVE

SAVE ⟨*nombre de fichero*⟩[,⟨*tipo de fichero*⟩][,⟨*parámetros binarios*⟩]

```
SAVE "fichero.xyz"
```

graba el fichero en modo de BASIC no protegido normal.

```
SAVE "fichero.xyz",P
```

graba el fichero en modo de BASIC protegido.

```
SAVE "fichero.xyz",A
```

graba el fichero en modo ASCII.

```
SAVE "fichero.xyz",B,8000,3000,8001
```

graba el fichero en modo binario. En este ejemplo, graba la zona de memoria del ordenador que empieza en la dirección **8000** y tiene una extensión de **3000** bytes; el punto de entrada opcional es **8001**.

Orden. Graba en cinta el programa actualmente residente en la memoria. Un fichero binario consiste en una zona de memoria copiada en la cinta. Los parámetros son en este caso:

⟨*dirección inicial*⟩,⟨*longitud del fichero*⟩[,⟨*punto de entrada*⟩]

La memoria dedicada a la pantalla se puede grabar en forma de fichero binario. Es lo que se denomina 'volcado de la pantalla'; la orden necesaria es:

```
SAVE "pantalla",B,&C000,&4000
```

El fichero se puede volver a cargar con la orden:

```
LOAD "pantalla",
```

Si el primer carácter del nombre del fichero es !, BASIC suprime tanto este carácter como todos los mensajes que de otra forma generarían el sistema de cinta.

Palabras clave asociadas: **CHAIN, CHAIN MERGE, LOAD, MERGE, RUN.**

SGN

SGN(⟨*expresión numérica*⟩)

```
10 FOR n=200 TO -200 STEP-20
20 PRINT "SGN da";
30 PRINT SGN(n);"cuando el numero es";n
40 NEXT
run
```

Función. Determina el signo de la ⟨expresión numérica⟩. Da el valor −1 si la ⟨expresión numérica⟩ es menor que cero; el valor 0 si es igual a 0; y el valor 1 si es mayor que cero.

Palabras clave asociadas: **ABS**.

SIN

SIN(⟨*expresión numérica*⟩**)**

```
10 CLS:DEG:ORIGIN 0,200
20 FOR n=0 TO 720
30 y=SIN(n)
40 PLOT n*640/720,198*y:NEXT
50 GOTO 50
run
```

Función. Calcula el seno de la ⟨expresión numérica⟩.

Obsérvese que se puede utilizar **DEG** y **RAD** para hacer que el ordenador entienda los arcos como dados en grados o en radianes, respectivamente.

Palabras clave asociadas: **ATN, DEG, COS, RAD, TAN**.

SOUND

SOUND ⟨*situación de canales*⟩,⟨*periodo de tono*⟩[,⟨*duración*⟩[,⟨*volumen*⟩
[,⟨*envolvente de volumen*⟩[,⟨*envolvente de tono*⟩[,⟨*ruido*⟩]]]]]

```
10 FOR z=0 TO 4095
20 SOUND 1,z,1,12
30 NEXT
run
```

Orden. Programa la generación de un sonido. Los parámetros son los siguientes:

Parámetro 1: ⟨situación de canales⟩

El parámetro ⟨situación de canales⟩ debe tener un valor comprendido entre 1 y 255. El significado de cada uno de los bits es como sigue:

Bit 0 (decimal 1)	enviar sonido al canal A (bit menos significativo)
Bit 1 (decimal 2)	enviar sonido al canal B
Bit 2 (decimal 4)	enviar sonido al canal C
Bit 3 (decimal 8)	sincronización con el canal A
Bit 4 (decimal 16	sincronización con el canal B
Bit 5 (decimal 32)	sincronización con el canal C
Bit 6 (decimal 64)	retención del canal de sonido
Bit 7 (decimal 128)	borrado del canal de sonido (bit más significativo)

Por ejemplo, el valor 68 del parámetro ⟨situación de canales⟩ especifica:

Enviar el sonido al canal C (4) y retenerlo (64).

Parámetro 2: ⟨periodo de tono⟩

Este parámetro define el tono del sonido; es decir, identifica la nota (DO, RE, MI, FA, etc.). Cada nota tiene un número de identificación, que es el ⟨periodo de tono⟩. Véase el capítulo titulado 'Para su referencia . . .'.

Parámetro 3: ⟨duración⟩

Este parámetro especifica la duración del sonido. Las unidades son de centésimas de segundo (0.01 s). Si no se especifica duración, el ordenador toma el valor implícito para este parámetro, que es 20 (un quinto de segundo).

Si el valor de ⟨duración⟩ es cero, el sonido dura hasta el final de la envolvente de volumen especificada.

Si el valor de ⟨duración⟩ es negativo, la envolvente de volumen especificada se repite **ABS**(⟨duración⟩) veces.

Parámetro 4: ⟨volumen⟩

Este parámetro especifica el volumen inicial de la nota. El margen de valores va de 0 a 15. El ⟨volumen⟩ 0 es el mínimo; el 15 es el máximo. Si no se especifica ⟨volumen⟩, el ordenador toma el implícito, que es 12.

Parámetro 5: ⟨envolvente de volumen⟩

Se puede hacer que el volumen de la nota no sea constante, sino que varíe con el tiempo mientras la nota está sonando. Para especificar la forma de variación del volumen con el tiempo se utiliza la orden **ENV**. De hecho, se pueden crear hasta 15 envolventes de volumen diferentes, con números de referencia del 1 al 15. El parámetro ⟨envolvente de volumen⟩ invoca una envolvente de volumen para su utilización por **SOUND**.

Véase la descripción de la orden **ENV**.

Parámetro 6: ⟨envolvente de tono⟩

Se puede hacer que el tono de una nota no sea constante, sino que varíe con el tiempo mientras la nota está sonando. Para especificar la forma de variación del tono con el tiempo se utiliza la orden **ENT**. De hecho, se pueden crear hasta 15 envolventes de tono diferente, con números de referencia del 1 al 15. El parámetro ⟨envolvente de tono⟩ invoca una envolvente de tono para su utilización por **SOUND**. Si en la definición de **ENT** se ha puesto un número negativo como número de envolvente, en la instrucción **SOUND** se debe poner el valor absoluto de ese número como ⟨envolvente de tono⟩.

Véase la descripción de la orden **ENT**.

Parámetro 7: ⟨ruido⟩

Se puede añadir ruido al sonido: la intensidad del ruido se controla con el parámetro ⟨ruido⟩. El margen de valores va de 0 (ruido apagado) hasta 31.

En la parte 2 del capítulo 'Cuando usted guste . . .' daremos más amplia información sobre los sonidos.

Palabras clave asociadas: **ENT, ENV, ON SQ GOSUB, RELEASE, SQ.**

SPACE$

SPACE$(⟨*expresión entera*⟩**)**

```
10 MODE 1
20 PRINT "Pongamos 5 espacios entre uno";
30 PRINT SPACE$(5);
40 PRINT "y otro"
run
```

Función. Crea una cadena literal consistente en el número de espacios especificado por ⟨expresión entera⟩ (margen de 0 a 255).

Palabras clave asociadas: **SPC, STRING$, TAB.**

SPC

Véase **PRINT SPC.**

SPEED INK

SPEED INK ⟨*periodo 1*⟩,⟨*periodo 2*⟩

```
10 BORDER 7,18
20 FOR i=30 TO 1 STEP-1
30 SPEED INK i,i
40 FOR t=1 TO 700:NEXT t,i
run
```

Orden. Establece la velocidad de alternancia entre los dos colores de tinta especificados mediante **INK** o **BORDER**. El primer parámetro, ⟨periodo 1⟩, especifica el tiempo para el primer color, en unidades de 0.02 s (cincuentavos de segundo); ⟨periodo 2⟩ especifica el tiempo para el segundo color.

Téngase cuidado al elegir colores y velocidades para evitar efectos hipnotizantes.

Palabras clave asociadas: **BORDER, INK.**

SPEED KEY

SPEED KEY ⟨*demora de arranque*⟩,⟨*periodo de repetición*⟩

```
10 CLS:FOR k=7 TO 1 STEP-2
20 PRINT "Escriba su nombre y pulse [INTRO]"
30 SPEED KEY k,k
40 LINE INPUT a$:NEXT
50 PRINT "Es un nombre muy interesante"
run
```

Orden. Establece las características de repetición de las teclas. El parámetro ⟨demora de arranque⟩ especifica el tiempo, en unidades e 0.02 s (cincuentavos de segundo) que ha de estar pulsada la tecla antes de que empiece a repetirse su acción. El parámetro ⟨periodo de repetición⟩ especifica el tiempo que debe transcurrir entre cada dos repeticiones sucesivas.

SPEED KEY solamente actúa sobre las teclas que son de repetición implícitamente o que han sido definidas como tales mediante **KEY DEF**.

Cuando se vaya a utilizar valores muy bajos del parámetro ⟨demora de arranque⟩, es aconsejable definir previamente una de las teclas numéricas para que devuelva el teclado a la situación implícita, **SPEED KEY 30,2**. La orden

```
KEY 0,"SPEED KEY 30,2"+CHR$(13)
```

restablece los valores implícitos de los parámetros de **SPEED KEY** en cuanto se pulsa la tecla del **0** del teclado numérico.

Palabras clave asociadas: **KEY DEF**.

SPEED WRITE

SPEED WRITE ⟨*expresión entera*⟩

```
SPEED WRITE 1
```

Orden. Establece la velocidad de transferencia de datos a la cinta. Las velocidades son: 2000 bits por segundo, si ⟨expresión entera⟩ vale 1, y 1000 bits por segundo si ⟨expresión entera⟩ vale 0. Esta última es la velocidad implícita. Para leer una cinta no es necesario saber a qué velocidad ha sido grabada, pues el ordenador selecciona automáticamente la velocidad correcta de lectura.

La máxima fiabilidad en la transferencia de datos se obtiene con la velocidad mínima, que además es la implícita: **SPEED WRITE 0**.

Palabras clave asociadas: **OPENOUT**, **SAVE**.

SQ

SQ(⟨*canal*⟩**)**

```
10 SOUND 65,100,100
20 PRINT SQ(1)
run
 67
```

Función. Genera un número que informa del estado de la cola de sonido del ⟨canal⟩ especificado. El parámetro puede tener uno de los siguientes valores:

1 para el canal A
2 para el canal B
4 para el canal C

La función **SQ** produce un número que se interpreta, según sea el valor de sus bits, de la forma siguiente:

Bits 0, 1 y 2:		número de huecos libres en la cola
Bits 3, 4 y 5:		estado de sincronización de la primera nota de la cola
Bit 6	:	la primera nota está retenida
Bit 7	:	el canal está activo en la actualidad

donde el bit 0 es el menos significativo y el bit 7 el más significativo.

Como se puede apreciar, si el bit 6 está a 1, el bit 7 tiene que estar a cero, y viceversa. Análogamente, si alguno de los bits 3, 4 y 5 está a 1, los bits 6 y 7 tienen que estar a cero.

En la parte 2 del capítulo 'Cuando usted guste . . .' daremos más información acerca de los sonidos.

Palabras clave asociadas: **ON SQ GOSUB**, **SOUND**.

SQR

SQR(⟨*expresión numérica*⟩**)**

```
PRINT SQR(9)
 3
```

Función. Da el valor de la raíz cuadrada de la ⟨expresión numérica⟩.

STEP

Véase **FOR**.

STOP

STOP

```
10 FOR n=1 TO 30:PRINT n:NEXT
20 STOP
30 FOR n=31 TO 60:PRINT n:NEXT
run
cont
```

Orden. Detiene la ejecución del programa, pero deja BASIC en un estado tal, que el programa puede ser reanudado mediante **CONT**. **STOP** se utiliza para establecer puntos de parada en la depuración de programas.

Palabras clave asociadas: **CONT**, **END**.

STR$

STR$(⟨*expresión numérica*⟩**)**

```
10 a=&FF :REM 255 en hexadecimal
20 b=&X1111 :REM 15 en binario
30 c$="***"
40 PRINT c$+STR$(a+b)+c$
run
*** 270***
```

Función. Convierte la ⟨expresión numérica⟩ en una cadena literal formada por los mismos dígitos decimales.

Palabras clave asociadas: **BIN$**, **HEX$**, **VAL**.

STRING$

STRING$(⟨*longitud*⟩,⟨*especificador del carácter*⟩**)**

```
PRINT STRING$(40,"*")
****************************************
```

Función. Da una cadena literal que consiste en el carácter especificado repetido tantas veces como indique ⟨longitud⟩ (margen de 0 a 255). Nótese que en el ejemplo anterior se podría haber puesto también:

```
PRINT STRING$(40,42)
****************************************
```

donde el ⟨especificador del carácter⟩, el número 42, es el código ASCII del asterisco.

Palabras clave asociadas: **SPACE$**.

SWAP

Véase **WINDOW SWAP**.

SYMBOL

SYMBOL ⟨*número del carácter*⟩,⟨*lista de:*⟨*fila*⟩

```
10 MODE 1:SYMBOL AFTER 105
20 fila1=255:REM 11111111 en binario
30 fila2=129:REM 10000001 en binario
40 fila3=189:REM 10111101 en binario
50 fila4=153:REM 10011001 en binario
60 fila5=153:REM 10011001 en binario
70 fila6=189:REM 10111101 en binario
80 fila7=129:REM 10000001 en binario
90 fila8=255:REM 11111111 en binario
100 PRINT "La linea 110 redefine la letra i (105). Teclee varias
    veces esa letra y luego liste el programa".
110 SYMBOL 105,fila1,fila2,fila3,fila4,fila5,fila6,fila7,fila8
run
```

Orden. Redefine la forma de un carácter. El margen para todos los parámetros es de 0 a 255.

Con objeto de reservar espacio en la memoria del ordenador para la descripción del carácter, es necesario ejecutar previamente la orden:

```
SYMBOL AFTER x
```

donde x debe ser igual o menor que el número del carácter que se va a definir.

A continuación se emite la orden **SYMBOL** seguida, en primer lugar, por el número del carácter.

El carácter puede ser directamente accesible por el teclado; en cualquier caso se lo puede escribir mediante la orden

```
PRINT CHR$(x)
```

Después de **SYMBOL x** viene una sucesión de hasta 8 parámetros, cada uno de los cuales describe una fila de la matriz del carácter. El margen de estos parámetros es de 0 a 255. La representación binaria de cada parámetro indica la situación de 'encendido' o 'apagado' de cada uno de los puntos de la fila correspondiente.

Por ejemplo, si el primero de los ochos parámetros vale 1, la fila superior del carácter tiene la representación binaria 00000001. Cada 1 indica que el correspondiente punto del carácter se ilumina con el color de PEN; cada 0 indica que el correspondiente punto no será visible, pues se lo escribe con el color de PAPER. Así pues, la primera fila de este carácter consiste en un punto en el extremo superior derecho. Los siguientes parámetros son **3,7,15,31,63,0,0**. La representación binaria de los ocho parámetros es como sigue:

parámetro (fila) 1: 00000001 en binario (1 en decimal)
parámetro (fila) 2: 00000011 en binario (3 en decimal)
parámetro (fila) 3: 00000111 en binario (7 en decimal)
parámetro (fila) 4: 00001111 en binario (15 en decimal)
parámetro (fila) 5: 00011111 en binario (31 en decimal)
parámetro (fila) 6: 00111111 en binario (63 en decimal)
parámetro (fila) 7: 00000000 en binario (0 en decimal)
parámetro (fila) 8: 00000000 en binario (0 en decimal)

Observando el conjunto de las representaciones binarias podemos hacernos una idea del aspecto que tendrá el carácter. Asignemos estos parámetros al carácter número 255 mediante la orden:

SYMBOL 255,1,3,7,15,31,63,0,0

Al ser **0** los dos últimos, podemos omitirlos:

SYMBOL 255,1,3,7,15,31,63

Los parámetros se pueden introducir en forma binaria, con lo que se ahorra el trabajo de convertirlos previamente a la forma decimal. (Recuérdese que el prefijo para números binarios es **&X**.) Por ejemplo:

SYMBOL 255,&X00000001,&X00000011,&X00000111,&X00001111,&X00011111, &X00111111

Finalmente, para ver el resultado de la definición, ejecute la orden

PRINT CHR$(255)

Si se redefine uno de los caracteres accesibles por el teclado, el nuevo carácter aparecerá en la pantalla cada vez que se pulse la tecla correspondiente o se lo escriba con **PRINT CHR$( )**. Por otra parte, BASIC reconoce la forma del nuevo carácter cuando trata de leerlo en la pantalla, y no lo rechaza por incomprensible.

En la parte 2 del capítulo 'Cuando usted guste . . .' daremos más información sobre la redefinición de caracteres.

Palabras clave asociadas: **HIMEM**, **MEMORY**, **SYMBOL AFTER**.

SYMBOL AFTER

SYMBOL AFTER ⟨*expresión entera*⟩

```
10 CLS
20 SYMBOL AFTER 115
30 PRINT "La linea 40 redefine la letra s ";
40 SYMBOL 115,0,56,64,64,48,8,8,112
50 PRINT "a s"
60 PRINT "Cancele esta definicion de s"
70 PRINT "con la orden SYMBOL AFTER 240"
run
```

Orden. Establece el número de caracteres redefinibles por el usuario (en el margen de 0 a 256). El número implícito es 240, lo que deja 16 caracteres (del 240 al 255) definibles por el usuario. Si ⟨expresión entera⟩ es 32, todos los caracteres, del 32 al 255, son definibles. **SYMBOL AFTER 256** impide la redefinición de caracteres.

Cada vez que se ejecuta una orden **SYMBOL AFTER**, quedan anuladas todas las redefiniciones y la forma de todos los caracteres vuelve a ser la implícita.

La orden **SYMBOL AFTER** *no* se puede ejecutar después de modificar el valor de **HIMEM**, lo cual puede ocurrir como resultado de una orden **MEMORY**, o bien al abrir un tampón para la grabación o lectura de un fichero con **OPENOUT** u **OPENIN**. En tales circunstancias se provoca el mensaje '**Improper argument**' ('argumento inadecuado', error número 5), a menos que la situación anterior fuera **SYMBOL AFTER 256**.

En la parte 2 del capítulo 'Cuando usted guste . . .' daremos más información sobre la redefinición de caracteres.

Palabras clave asociadas: **HIMEM**, **MEMORY**, **SYMBOL**.

TAB

Véase **PRINT TAB**.

TAG

TAG [#⟨*expresión de número de canal*⟩]

```
10 INPUT "Escriba su nombre: ",a$:CLS
20 PRINT "Se mueve usted mucho, ";a$
30 TAG
40 x=LEN(a$)*17:y=50+RND*300:MOVE -x,y
50 FOR f=-x TO 640 STEP RND*7+3
60 MOVE f,y:PRINT " ";a$;:CALL &BD19:NEXT
```

```
70 FOR b=640 TO -x STEP-RND*7+3
80 MOVE b,y:PRINT a$;" ";:CALL &BD19:NEXT
90 GOTO 40
run
```

Orden. Hace que el texto enviado al canal especificado sea escrito en la pantalla gráfica, en la posición actual del cursor. De esta forma se puede mezclar texto con gráficos, o mover los caracteres pixel a pixel. Si se omite la ⟨expresión de número de canal⟩, se supone el **#0**.

El punto superior izquierdo del carácter se conecta al cursor gráfico. Si se omite el signo de punto y coma al final de la instrucción **PRINT**, los códigos de retorno del carro y de avance de línea se visualizan como caracteres gráficos.

Si se ha aplicado **TAG** al canal **#0**, BASIC desactiva **TAG** al volver a modo directo.

Palabras clave asociadas: **TAGOFF**.

TAGOFF

TAGOFF [#⟨*expresión de número de canal*⟩]

```
10 MODE 2:TAG :REM texto ligado al cursor grafico
20 periodo=1986:FOR x=1 TO 640 STEP 70
30 MOVE x,400:DRAWR 0,-350
40 periodo=periodo+1:PRINT periodo;:NEXT
50 TAGOFF :REM texto desconectado de cursor gráfico
60 LOCATE 34,25:PRINT "Datos anuales"
70 GOTO 70
run
```

Orden. Cancela la orden **TAG** para el canal especificado (o para el **#0**, si no se especifica otro) y redirige el texto a la posición que tenía el cursor de texto antes de ejecutarse **TAG**.

Palabras clave asociadas: **TAG**.

TAN

TAN(⟨*expresión numérica*⟩**)**

```
PRINT TAN(45)
 1.61977519
```

Función. Calcula la tangente de la ⟨expresión numérica⟩, que debe tener un valor comprendido entre −20000 y +20000.

Obsérvese que se puede utilizar **DEG** y **RAD** para hacer que el ordenador entienda los arcos como dados en grados o en radianes, respectivamente.

Palabras clave asociadas: **ATN**, **COS**, **DEG**, **RAD**, **SIN**.

TEST

TEST(⟨*coordenada x*⟩,⟨*coordenada y*⟩**)**

```
10 CLS
20 PRINT "Estamos utilizando la pluma numero";
30 PRINT TEST(10,386)
40 PRINT "Cambie la pluma y el modo ";
50 PRINT ". . . y ejecute otra vez el programa."
run
```

Función. Traslada el cursor gráfico al punto de coordenadas absolutas especificadas e informa del valor de la tinta presente en esa posición.

Palabras clave asociadas: **DRAW, DRAWR, MOVE, MOVER, PLOT, PLOTR, TESTR, XPOS, YPOS.**

TESTR

TESTR(⟨*incremento de x*⟩,⟨*incremento de y*⟩**)**

```
10 MODE 0:FOR x=1 TO 15:LOCATE 1,x
20 PEN x:PRINT STRING$(10,143);:NEXT
30 MOVE 200,400:PEN 1
40 FOR n=1 TO 23:LOCATE 12,n
50 PRINT "pen";TESTR(0,-16):NEXT
run
```

Función. Lleva el cursor gráfico a a un punto cuya posición, referida a la posición actual, viene dada por los parámetros ⟨incremento de x⟩ e ⟨incremento de y⟩, e informa del valor de la tinta presente en ese punto.

Palabras clave asociadas: **DRAW, DRAWR, MOVE, MOVER, PLOT, PLOTR, TEST, XPOS, YPOS.**

THEN

Véase **IF**.

TIME

TIME

```
10 CLS:REM reloj
20 INPUT "Hora: ",hora
30 INPUT "Minuto: ",minuto
40 INPUT "Segundo: ",segundo
50 CLS:referencia=INT(TIME/300)
60 WHILE hora<13
70 WHILE minuto<60
80 WHILE tiempo<60
90 tiempo=(INT(TIME/300)-referencia)+segundo
100 LOCATE 1,1
110 PRINT USING "## ";hora,minuto,tiempo
120 WEND
130 tiempo=0:segundo=0:minuto=minuto+1
140 GOTO 50
150 WEND
160 minuto=0:hora=hora+1
170 WEND
180 hora=1
190 GOTO 60
```

Función. Da el tiempo transcurrido desde el momento en que se encendió o reinicializó la máquina por última vez (excluyendo el dedicado a escribir o leer en el disco), en unidades de 1/300 s. Expresado en segundos, el tiempo es, pues, **TIME/300**.

Palabras clave asociadas: **AFTER**, **EVERY**, **WEND**, **WHILE**.

TO

Véase **FOR**.

TROFF
TRON

TROFF
TRON

```
10 TROFF:PRINT:PRINT "TROFF"
20 FOR n=1 TO 8
30 PRINT "Programa funcionando":NEXT
40 IF f=1 THEN END
50 TRON:PRINT:PRINT "TRON"
60 f=1:GOTO 20
run
```

Orden. Sigue la 'traza' del programa escribiendo los números de cada línea antes de ejecutarla. Los números de línea aparecen entre corchetes [ ].

TRON activa este modo; **TROFF** lo desactiva.

TRON es particularmente útil para estudiar la secuencia de números de línea anterior al momento en que se produce un error.

Palabras clave asociadas: **RUN**.

UNT

UNT(⟨*expresión de dirección de memoria*⟩**)**

```
PRINT UNT(&FF66)
-154
```

Orden. Da un entero, comprendido entre −32768 y +32767, que es el equivalente en 'complemento a dos' al valor de ⟨expresión de dirección de memoria⟩.

Palabras clave asociadas: **CINT**, **FIX**, **INT**, **ROUND**.

UPPER$

UPPER$(⟨*expresión literal*⟩**)**

```
10 CLS:a$="mire como crecen!"
20 PRINT UPPER$(a$)
run
```

Función. Da una cadena literal que es copia de la ⟨expresión literal⟩ especificada, pero convirtiendo las letras minúsculas en mayúsculas y dejando inalterados los restantes caracteres. Esta función es útil en el proceso de datos que puedan contener minúsculas mezcladas con mayúsculas.

Palabras clave asociadas: **LOWER$**.

USING

Véase **PRINT USING**.

VAL

VAL(⟨*expresión literal*⟩**)**

```
10 CLS:PRINT "Ya he aprendido la tabla de multiplicar"
20 PRINT:PRINT "Pulse una tecla del 1 al 9:"
30 a$=INKEY$:IF a$="" THEN 30
40 n=VAL(a$):IF n<1 OR n>9 THEN 30
50 FOR x=1 TO 10
60 PRINT n;"x";x;"=";n*x
70 NEXT:GOTO 20
run
```

Función. Da el valor numérico (incluyendo el signo y el punto decimal) de los primeros caracteres de la ⟨expresión literal⟩ que sean dígitos.

Si el primer carácter no es un dígito, el valor generado es 0. Si el primer carácter es el signo menos o el punto decimal y va seguido de caracteres no numéricos, se provoca el mensaje **'Type mismatch'** ('incongruencia de tipos', error número 13).

Palabras clave asociadas: **STR$**.

VPOS

VPOS(#⟨*expresión de número de canal*⟩**)**

```
PRINT VPOS(#3)
```

Función. Da la posición vertical del cursor de texto, referida al extremo superior de la ventana. Es necesario especificar la ⟨expresión de número de canal⟩, pues el ordenador no da por supuesto el **#0**.

Palabras clave asociadas: **POS**.

WAIT

WAIT ⟨*número de puerta*⟩,⟨*máscara*⟩[,⟨*inversión*⟩]

```
WAIT &FF34,20,25
```

Orden. Espera hasta que por la puerta de entrada/salida especificada por ⟨número de puerta⟩ se reciba un determinado valor. BASIC lee incesantemente la puerta. Con el valor recibido y el parámetro ⟨inversión⟩ se realiza la operación lógica **XOR**; con este resultado y el parámetro ⟨máscara⟩ se realiza la operación lógica **AND**; el proceso se repite hasta que el resultado final sea distinto de cero.

BASIC espera indefinidamente hasta que se cumple la condición requerida.

Esta orden no debe ser utilizada por quien no sepa cuáles son sus efectos.

Palabras clave asociadas: **INP**, **OUT**.

WEND

WEND

```
WEND
```

Orden. Señala el final del grupo de instrucciones cuya ejecución se repite en un bucle del tipo **WHILE**. BASIC determina automáticamente a qué **WHILE** corresponde cada **WEND**.

Palabras clave asociadas: **TIME**, **WHILE**.

WHILE

WHILE ⟨*expresión lógica*⟩

```
10 CLS:PRINT "Temporizador de 10 segundos":t=TIME
20 WHILE TIME<t+3000
30 SOUND 1,0,100,15
40 WEND:SOUND 129,40,30,15
run
```

Orden. Repite la ejecución de un grupo de instrucciones mientras se cumpla cierta condición. La orden **WHILE** señala el principio del bucle y especifica la condición lógica mediante la ⟨expresión lógica⟩.

Palabras clave asociadas: **TIME**, **WEND**.

WIDTH

WIDTH ⟨*expresión entera*⟩

```
WIDTH 40
```

Orden. Informa a BASIC de cuántos caractes por línea se deben enviar a la impresora. BASIC envía a la impresora los códigos de retorno del carro y avance de línea en el momento adecuado.

El valor implícito es **132**.

La orden **WIDTH 255** suprime los códigos de retorno del carro y avance de línea, dejando que el número de caracteres por línea sea totalmente controlado por la impresora. En todo caso, la orden **PRINT** envía a la impresora esos dos códigos siempre que no se incluya el signo de punto y coma al final.

Palabras clave asociadas: **POS**.

WINDOW

WINDOW [#⟨*expresión de número de canal*⟩,]⟨*izquierda*⟩,⟨*derecha*⟩,⟨*arriba*⟩,⟨*abajo*⟩

```
10 MODE 0:BORDER 0:REM carta de ajuste
20 INK 0,0:INK 1,25:INK 2,23:INK 3,21
30 INK 4,17:INK 5,6:INK 6,2:INK 7,26
40 PAPER 0:CLS
50 PAPER 1:WINDOW 2,4,1,18:CLS
60 PAPER 2:WINDOW 5,7,1,18:CLS
70 PAPER 3:WINDOW 8,10,1,18:CLS
80 PAPER 4:WINDOW 11,13,1,18:CLS
90 PAPER 5:WINDOW 14,16,1,18:CLS
100 PAPER 6:WINDOW 17,19,1,18:CLS
110 PAPER 7:WINDOW 2,19,19,25:CLS
120 GOTO 120
run
```

Orden. Especifica las dimensiones y situación de una ventana de texto. Los valores de los parámetros ⟨izquierda⟩, ⟨derecha⟩, ⟨arriba⟩ y ⟨abajo⟩ indican los números extremos de columna y de fila, y deben ser consistentes con el modo de pantalla actual.

Si no se especifica la ⟨expresión de número de canal⟩, BASIC supone el **#0**.

En la parte 2 del capítulo titulado 'Cuando usted guste . . .' daremos más información sobre ventanas de texto.

Palabras clave asociadas: **WINDOW SWAP**.

WINDOW SWAP

WINDOW SWAP ⟨*expresión de número de canal*⟩,⟨*expresión de número de canal*⟩

```
10 MODE 1:INK 1,24:INK 2,9:INK 3,6
20 WINDOW 21,40,13,25:PAPER 3
30 WINDOW #1,1,20,1,12:PAPER #1,2
40 CLS:PRINT " Ventana numero 0"
50 CLS #1:PRINT #1," Ventana numero 1"
60 LOCATE 1,6
```

```
70 PRINT " Ventana roja (0)";SPC(2)
80 LOCATE #1,1,6
90 PRINT #1," Ventana verde (1)"
100 FOR t=1 TO 1000:NEXT
110 WINDOW SWAP 0,1:GOTO 60
run
```

Orden. Intercambia las características de las dos ventanas de texto especificadas por las ⟨expresiones de número de canal⟩.

Se deben especificar los dos números de canal, pero en este caso **no** deben ir precedidos del signo #.

En la parte 2 del capítulo titulado 'Cuando usted guste . . .' daremos más información sobre ventanas de texto.

Palabras clave asociadas: **WINDOW**.

WRITE

WRITE [#⟨*expresión de número de canal*⟩,][⟨*lista de datos*⟩]

```
10 REM escribir variables en la cinta
20 INPUT "Escriba un numero: ",a
30 INPUT "Escriba una palabra: ",a$
40 OPENOUT "!datos"
50 WRITE #9,a,a$
60 CLOSEOUT:PRINT "Datos grabados en cinta"
run
```

Orden. Envía los valores de los elementos de la ⟨lista de datos⟩ al canal especificado por la ⟨expresión de número de canal⟩. Los elementos irán separados por comas; las cadenas literales irán entre comillas.

En el ejemplo anterior, los valores captados por el teclado son enviados al canal **#9** (cinta).

Para leer esos valores después de grabarlos en la cinta, se necesita un programa tal como el siguiente:

```
10 REM leer variables en la cinta
20 OPENIN "!datos":INPUT #9,a,a$
30 CLOSEIN:PRINT "Los dos valores son:"
40 PRINT:PRINT a,a$
run
```

Palabras clave asociadas: **INPUT**, **LINE INPUT**.

XOR

⟨*argumento*⟩ **XOR** ⟨*argumento*⟩

```
IF "juan"<"pepe" XOR "perro">"gato" THEN PRINT "verdadero" ELSE PRINT "falso"
falso
IF "pepe"<"juan" XOR "gato">"perro" THEN PRINT "verdadero" ELSE PRINT "falso"
falso
IF "juan"<"pepe" XOR "gato">"perro"THEN PRINT "verdadero" ELSE PRINT "falso"
verdadero
...
PRINT 1 XOR 1
 0
PRINT 0 XOR 0
 0
PRINT 1 XOR 0
 1
```

Operador. Realiza la operación lógica 'o exclusivo' bit a bit entre enteros. El resultado es 1 a menos que ambos bits tengan el mismo valor.

Para más amplia información sobre operaciones lógicas, consulte la parte 2 del capítulo titulado 'Cuando usted guste ...'.

Palabras clave asociadas: **AND, OR, NOT**.

XPOS

XPOS

```
10 MODE 1:DRAW 320,200
20 PRINT "Posicion horizontal del cursor grafico =";
30 PRINT XPOS
run
```

Función. Da el valor de la posición horizontal (coordenada **x**) del cursor gráfico.

Palabras clave asociadas: **MOVE, MOVER, ORIGIN, YPOS**.

YPOS

YPOS

```
10 MODE 1:DRAW 320,200
20 PRINT "Posicion vertical del cursor grafico =";
30 PRINT YPOS
run
```

Función. Da el valor de la posición vertical (coordenada **y**) del cursor gráfico.

Palabras clave asociadas: **MOVE**, **MOVER**, **ORIGIN**, **XPOS**.

ZONE

ZONE ⟨*expresión entera*⟩

```
10 CLS:FOR z=2 TO 20
20 ZONE z
30 PRINT "X","X ZONE =";z:NEXT
run
```

Orden. Cambia la anchura de la zona de escritura (especificada en las instrucciones **PRINT** cuando se utiliza una coma para separar elementos). La achura implícita es 13 columnas, pero se la puede cambiar al valor dado por la ⟨expresión entera⟩, cuyo margen es de 1 a 255.

Palabras clave asociadas: **PRINT**, **WIDTH**.

Capítulo 4
Para su referencia ...

Este capítulo contiene la mayor parte de la información que usted pueda necesitar para aprender a manejar el ordenador.

Temas tratados:

* Posiciones del cursor y códigos de control
* Interrupciones
* Caracteres ASCII y caracteres gráficos
* Esquemas del teclado
* Sonidos
* Mensajes de error
* Palabras reservadas de BASIC
* Plantillas
* Conexiones
* Impresoras
* Joysticks
* Ampliaciones del sistema residentes (RSX)
* Memoria

Para más completa información sobre BASIC y el firmware del CPC464, consúltense las publicaciones SOFT157 y SOFT158, respectivamente.

Parte 1: Posiciones del cursor y códigos ampliados de control en BASIC

En ciertas aplicaciones el cursor puede encontrarse fuera de la ventana de texto. Algunas operaciones fuerzan al cursor a ocupar una posición permitida antes de poder ser llevadas a cabo. Son las siguientes:

1. Escribir un carácter.
2. Exhibir el signo del cursor (el rectángulo).
3. Obedecer los códigos de control señalados con un asterisco en la tabla siguiente.

El proceso de forzar el cursor a ocupar una posición permitida consiste en lo siguiente:

1. Si el cursor está a la derecha del borde derecho de la ventana, salta al extremo izquierdo de la línea siguiente.
2. Si el cursor está a la izquierda del borde izquierdo de la ventana, salta al extremo derecho de la línea anterior.
3. Si el cursor está por encima del borde superior, el contenido de la ventana se desplaza una línea hacia abajo y el cursor aparece en la primera línea.
4. Si el cursor está por debajo del borde inferior, el contenido de la ventana se desplaza una línea hacia arriba y el cursor aparece en la última línea.

Las comprobaciones y operaciones se realizan en el orden de la lista anterior. Las posiciones ilegales del cursor pueden ser cero o negativas, es decir, a la izquierda y por encima de la ventana.

Los códigos del margen 0 a 31 no producen ningún símbolo cuando se los envía a la pantalla, sino que son interpretados como *códigos de control* (y deben ser utilizados con cuidado). Algunos de ellos alteran el significado de uno o varios de los códigos siguientes, que hacen las funciones de parámetros del código original.

Si se envía un código de control a la pantalla gráfica, su efecto es dibujar un símbolo relacionado con la función que tendría si se lo generase mediante el teclado (v. g., &07 'BEL': CTRL G). Ejercerá su función de control si se lo envía a la pantalla mediante:

```
PRINT CHR$(&07)
```
o bien
```
PRINT "Ω"
```

donde el símbolo del "timbre" se obtiene pulsando CTRL G en el momento de teclear la instrucción **PRINT**.

Los códigos señalados con * en la lista siguiente fuerzan al cursor a ocupar una posición permitida en la ventana antes de ser obedecidos. Para cada código se da el número, en hexadecimal y en decimal, y la descripción de su función.

Códigos de control en BASIC

Valor		Nombre	Parámetro	Significado
&00	0	NUL		Sin efecto. Se lo ignora.
&01	1	SOH	0 a 255	Escribe el símbolo correspondiente al valor del parámetro, lo que permite exhibir los símbolos del 0 al 31.
&02	2	STX		Hace invisible el cursor de texto.
&03	3	ETX		Hace visible el cursor de texto. BASIC inhibe el cursor en todas las situaciones, salvo cuando está esperando una entrada por el teclado. Tal acción automática de BASIC predomina sobre este código de control.
&04	4	EOT	0 a 2	Establece el modo de pantalla. El parámetro se toma MOD 4. Equivale a la orden **MODE**.
&05	5	ENQ	0 a 255	Envía a la posición del cursor gráfico el carácter correspondiente al valor del parámetro.
&06	6	ACK		Habilita la pantalla de texto. (Véase &15 NAK.)
&07	7	BEL		Produce un pitido. Obsérvese que este código borra las colas de sonido.
&08	8	*BS		Hace retroceder el cursor una posición.
&09	9	*TAB		Hace avanzar el cursor una posición.
&0A	10	*LF		Hace bajar el cursor una línea.
&0B	11	*VT		Hace subir el cursor una línea.
&0C	12	FF		Borra la ventana de texto y lleva el cursor al extremo superior izquierdo. Equivale a la orden **CLS**.

Valor	Nombre	Parámetro	Significado
&0D 13	*CR		Lleva el cursor al extremo izquierdo de la línea actual.
&0E 14	SO	0 a 15	Establece la tinta asignada al papel. El parámetro se toma MOD 16. Equivale a la orden **PAPER**.
&0F 15	SI	0 a 15	Establece la tinta asignada a la pluma. El parámetro se toma MOD 16. Equivale a la orden **PEN**.
&10 16	*DLE		Borra el carácter que está en la posición del cursor. Rellena la célula del carácter con el color actual del papel.
&11 17	*DC1		Borra desde el extremo izquierdo de la ventana hasta la posición actual del cursor inclusive. Rellena las células afectadas con el color actual del papel.
&12 18	*DC2		Borra desde la posición actual del cursor inclusive hasta el extremo derecho de la ventana. Rellena las células afectadas con el color actual del papel.
&13 19	*DC3		Borra desde el principio de la ventana hasta la posición actual del cursor inclusive. Rellena las células afectadas con el color actual del papel.
&14 20	*DC4		Borra desde la posición actual del cursor inclusive hasta el final de la ventana. Rellena las células afectadas con el color actual del papel.
&15 21	NAK		Inhibe la pantalla de texto, la cual no reacciona ante nada que se le envía mientras no reciba el código ACK (&06 6)
&16 22	SYN	0 a 1	Se toma el parámetro MOD 2. Opción de transparencia: 0 desactiva, 1 activa.

Valor	Nombre	Parámetro	Significado
&17 23	ETB	0 a 3	Se toma el parámetro MOD 4. Establece el modo de tinta para gráficos: 0 normal 1 XOR 2 AND 3 OR
&18 24	CAN		Intercambia las tintas de pluma y papel.
&19 25	EM	0 a 255 0 a 255 0 a 255 0 a 255 0 a 255 0 a 255 0 a 255 0 a 255 0 a 255	Define la matriz de un carácter definible por el usuario. Equivale a la orden **SYMBOL**. Lleva nueve parámetros. El primero es el número del carácter; los ocho restantes, el diseño del carácter. El bit más significativo del primer byte corresponde al punto extremo superior izquierdo del carácter. El bit menos significativo del último byte corresponde al punto extremo inferior derecho del caracter.
&16 26	SUB	1 a 80 1 a 80 1 a 25 1 a 25	Define una ventana. Equivale a la orden **WINDOW**. Los dos primeros parámetros especifican los bordes izquierdo y derecho de la ventana; el menor de ellos se considera definitorio del borde izquierdo. Los dos últimos parámetros especifican los bordes superior e inferior de la ventana; el menor de ellos se considera definitorio del borde superior.
&1B 27	ESC		Sin efecto. Se lo ignora.
&1C 28	FS	0 a 15 0 a 31 0 a 31	Asigna dos colores a una tinta. Equivale a la orden **INK**. El primer parámetro (MOD 16) especifica el número de tinta; los dos siguientes (MOD 32), los colores deseados. (Los valores 27 a 31 no corresponden a colores definidos.)
&1D 29	GS	0 a 31 0 a 31	Asigna dos colores al borde. Equivale a la orden **BORDER**. Los dos parámetros (MOD 32) especifican los colores. (Los valores 27 a 31 no corresponden a colores definidos.)

Valor	Nombre	Parámetro	Significado
&1E 30	RS		Lleva el cursor al extremo superior izquierdo de la ventana.
&1F 31	US	1 a 80 1 a 25	Lleva el cursor a la posición específicada dentro de la ventana actual. Equivale a la orden **LOCATE**. El primer parámetro representa la columna; el segundo, la línea.

Parte 2: El sistema operativo

La gestión interna del CPC464 es realizada en tiempo real por un sofisticado sistema operativo. El sistema operativo es una especie de 'guardia de tráfico' que controla el flujo entre entradas y salidas.

Su función primordial es servir de intermediario (interfaz) entre BASIC y los circuitos electrónicos. Por ejemplo, cuando el programa exige que parpadeen las tintas, BASIC entrega al sistema operativo los parámetros y éste realiza el trabajo.

El sistema operativo de la máquina es lo que se suele denominar 'firmware'. Es un programa grabado en ROM y comprende las rutinas en código de máquina que son invocadas por las órdenes de alto nivel de BASIC.

Si se siente tentado a modificar el contenido de posiciones de memoria (con la orden **POKE**) o a invocar subrutinas del sistema operativo (con **CALL**), grabe el programa antes de que sea demasiado tarde.

El firmware del sistema operativo del CPC464 está descrito en la publicación SOFT 158 y queda fuera del alcance de este manual.

Si usted piensa programar en código de máquina, debe proveerse de un programa ensamblador. El DEVPAC de AMSOFT incluye un ensamblador de Z80 reubicable, con editor, desensamblador y monitor.

Parte 3: Interrupciones

El CPC464 utiliza ampliamente las interrupciones del Z80 para proporcionar un sistema operativo que incluye varias características multitarea, de las que son ejemplo las estructuras **AFTER** y **EVERY** descritas en otro lugar de este manual. El orden de prioridad de los temporizadores es el siguiente:

Break (ESC ESC) (detención del programa)
Temporizador 3
Temporizador 2 (y las tres colas de sonido)
Temporizador 1
Temporizador 0

Las interrupciones sólo se deben programar después de considerar las consecuencias de los posibles estados intermedios de las variables en el momento de la interrupción. La subrutina de interrupción debe evitar interacciones indeseadas con el estado de las variables del programa principal.

Las colas de sonidos tienen interrupciones independientes, de igual nivel de prioridad. Una vez iniciada una interrupción de sonido, ya no puede ser interrumpida por otra de sonido. Este mecanismo permite que las rutinas de interrupción de sonido puedan compartir variables sin estar sujetas a los riesgos mencionados.

Cuando se habilita una interrupción de cola de sonido (con **ON SQ GOSUB**), la interrupción se realiza inmediatamente si la cola de sonido del canal correspondiente no está llena; si lo está, la interrupción se producirá cuando termine el sonido actual y quede espacio libre en la cola. La acción de la interrupción inhibe el suceso, de modo que la subrutina debe ser rehabilitada si se la va a utilizar más tarde.

Las acciones de enviar un sonido y de examinar el estado de la cola inhiben también la interrupción de sonido.

Parte 4: Caracteres ASCII y caracteres gráficos en BASIC

ASCII

En la tabla siguiente se da la relación de los códigos y caracteres ASCII. Los números están en decimal, octal y hexadecimal. En las siguientes páginas se ilustra el diseño de los caracteres generados por el CPC464.

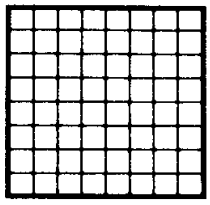
DEC	OCTAL	HEX	Caracteres ASCII
0	000	00	NUL (CTRL @)
1	001	01	SOH (CTRL A)
2	002	02	STX (CTRL B)
3	003	03	ETX (CTRL C)
4	004	04	EOT (CTRL D)
5	005	05	ENQ (CTRL E)
6	006	06	ACK (CTRL F)
7	007	07	BRL (CTRL G)
8	010	08	BS (CTRL H)
9	011	09	HT (CTRL I)
10	012	0A	LF (CTRL J)
11	013	0B	VT (CTRL K)
12	014	0C	FF (CTRL L)
13	015	0D	CR (CTRL M)
14	016	0E	SO (CTRL N)
15	017	0F	SI (CTRL O)
16	020	10	DLE (CTRL P)
17	021	11	DC1 (CTRL Q)
18	022	12	DC2 (CTRL R)
19	023	13	DC3 (CTRL S)
20	024	14	DC4 (CTRL T)
21	025	15	NAK (CTRL U)
22	026	16	SYN (CTRL V)
23	027	17	ETB (CTRL W)
24	030	18	CAN (CTRL X)
25	031	19	EM (CTRL Y)
26	032	1A	SUB (CTRL Z)
27	033	1B	ESC (CTRL [)
28	034	1C	FS (CTRL \)
29	035	1D	GS (CTRL])
30	036	1E	RS (CTRL ↑)
31	037	1F	US (CTRL 0)
32	040	20	SP
33	041	21	!
34	042	22	"
35	043	23	#
36	044	24	$
37	045	25	%
38	046	26	&
39	047	27	'
40	050	28	(
41	051	29	)
42	052	2A	*
43	053	2B	+
44	054	2C	,
45	055	2D	–
46	056	2E	.
47	057	2F	/
48	060	30	0
49	061	31	1

DEC	OCTAL	HEX	ASCII
50	062	32	2
51	063	33	3
52	064	34	4
53	065	35	5
54	066	36	6
55	067	37	7
56	070	38	8
57	071	39	9
58	072	3A	:
59	073	3B	;
60	074	3C	<
61	075	3D	=
62	076	3E	>
63	077	3F	?
64	100	40	@
65	101	41	A
66	102	42	B
67	103	43	C
68	104	44	D
69	105	45	E
70	106	46	F
71	107	47	G
72	110	48	H
73	111	49	I
74	112	4A	J
75	113	4B	K
76	114	4C	L
77	115	4D	M
78	116	4E	N
79	117	4F	O
80	120	50	P
81	121	51	Q
82	122	52	R
83	123	53	S
84	124	54	T
85	125	55	U
86	126	56	V
87	127	57	W
88	130	58	X
89	131	59	Y
90	132	5A	Z
91	133	5B	[
92	134	5C	\
93	135	5D	]
94	136	5E	↑
95	137	5F	_
96	140	60	`
97	141	61	a
98	142	62	b
99	143	53	c

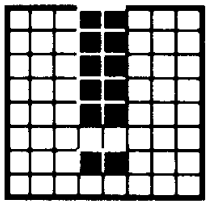
DEC	OCTAL	HEX	ASCII
100	144	64	d
101	145	65	e
102	146	66	f
103	147	67	g
104	150	68	h
105	151	69	i
106	152	6A	j
107	153	6B	k
108	154	6C	l
109	155	6D	m
110	156	6E	n
111	157	6E	o
112	160	70	p
113	161	71	q
114	162	72	r
115	163	73	s
116	164	74	t
117	165	75	u
118	166	76	v
119	167	77	w
120	170	78	x
121	171	79	y
122	172	7A	z
123	173	7B	{
124	174	7C	|
125	175	7D	}
126	176	7E	~

Conjunto de caracteres gráficos específicos de la máquina

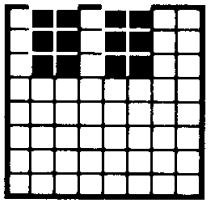
Los caracteres aquí reproducidos están representados en la matriz estándar de 8×8 en la que se basa la visualización de texto en la pantalla del CPC464. Los caracteres definidos por el usuario pueden ser agrupados para conseguir efectos especiales. Véase la sección 'Caracteres definidos por el usuario' en el capítulo 'Cuando usted guste . . .'.



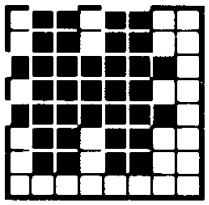
32 &H20
&X00100000



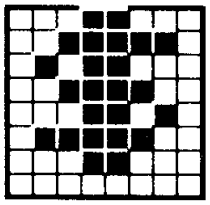
33
&H21
&X00100001



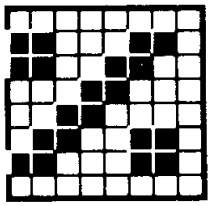
34
&H22
&X00100010



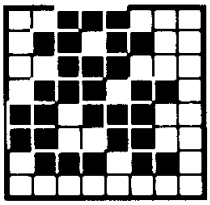
35
&H23
&X00100011



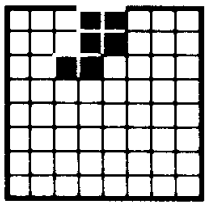
36
&H24
&X00100100



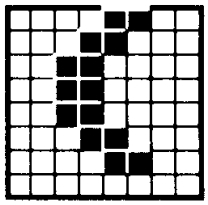
37
&H25
&X00100101



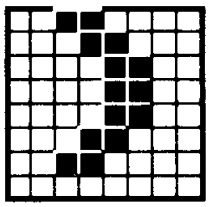
38
&H26
&X00100110



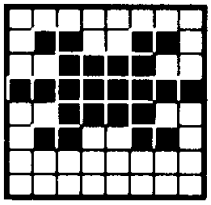
39
&H27
&X00100111



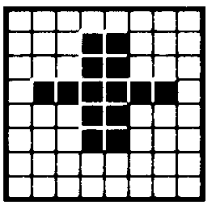
40
&H28
&X00101000



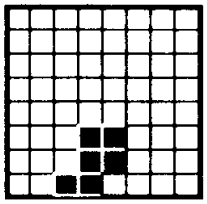
41
&H29
&X00101001



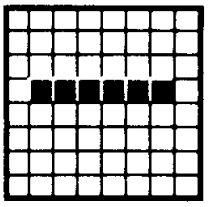
42
&H2A
&X00101010



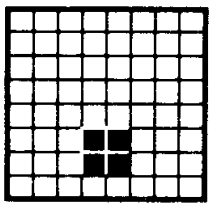
43
&H2B
&X00101011



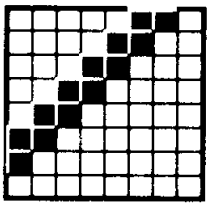
44
&H2C
&X00101100



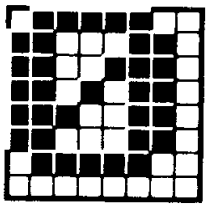
45
&H2D
&X00101101



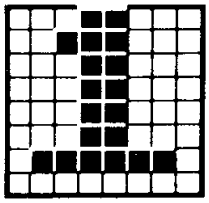
46
&H2E
&X00101110



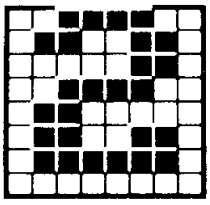
47
&H2F
&X00101111



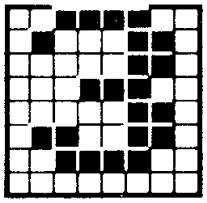
48
&H30
&X00110000



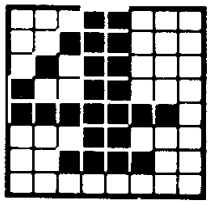
49
&H31
&X00110001



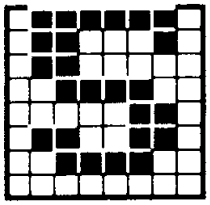
50
&H32
&X00110010



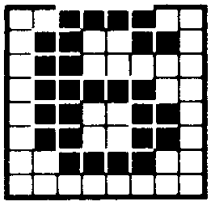
51
&H33
&X00110011



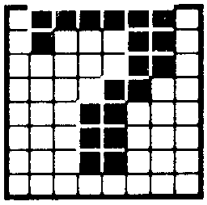
52
&H34
&X00110100



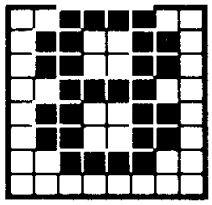
53
&H35
&X00110101



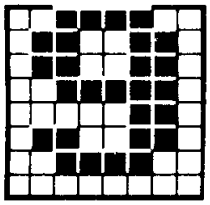
54
&H36
&X00110110



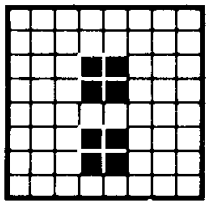
55
&H37
&X00110111



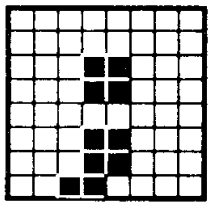
56
&H38
&X00111000



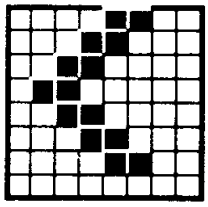
57
&H39
&X00111001



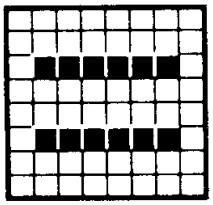
58
&H3A
&X00111010



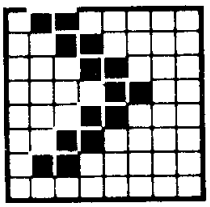
59
&H3B
&X00111011



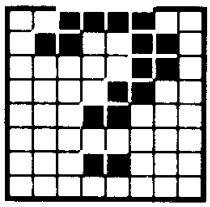
60
&H3C
&X00111100



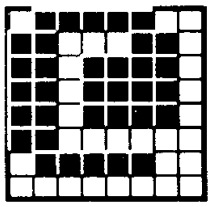
61
&H3D
&X00111101



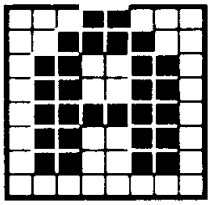
62
&H3E
&X00111110



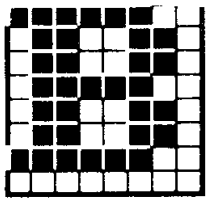
63
&H3F
&X00111111



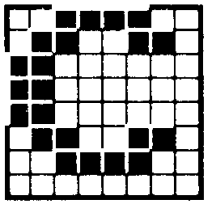
64
&H40
&X01000000



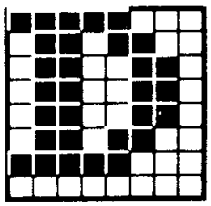
65
&H41
&X01000001



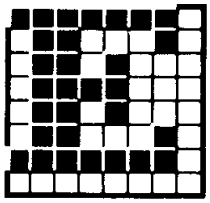
66
&H42
&X01000010



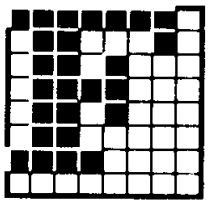
67
&H43
&X01000011



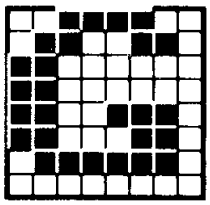
68
&H44
&X01000100



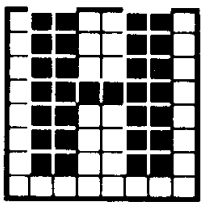
69
&H45
&X01000101



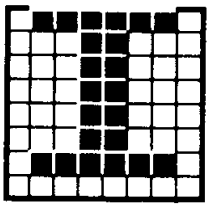
70
&H46
&X01000110



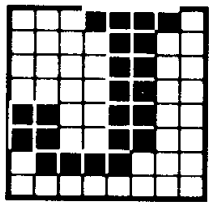
71
&H47
&X01000111



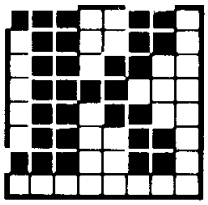
72
&H48
&X01001000



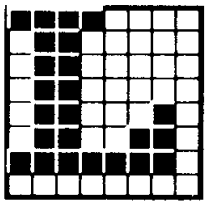
73
&H49
&X01001001



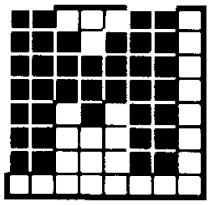
74
&H4A
&X01001010



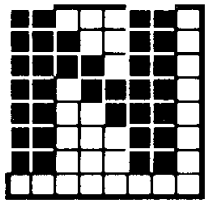
75
&H4B
&X01001011



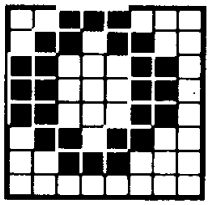
76
&H4C
&X01001100



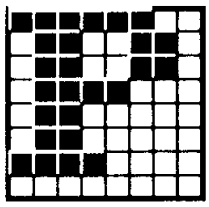
77
&H4D
&X01001101



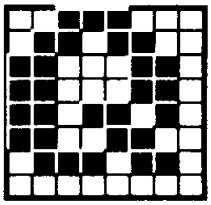
78
&H4E
&X01001110



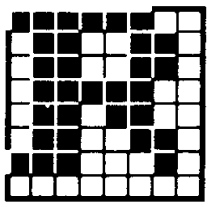
79
&H4F
&X01001111



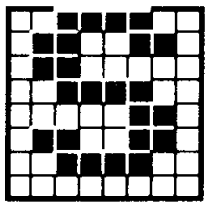
80
&H50
&X01010000



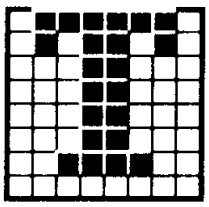
81
&H51
&X01010001



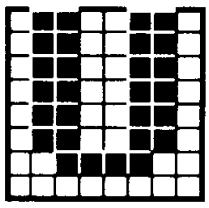
82
&H52
&X01010010



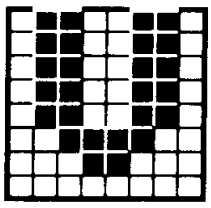
83
&H53
&X01010011



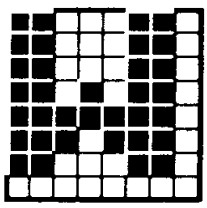
84
&H54
&X01010100



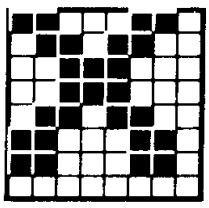
85
&H55
&X01010101



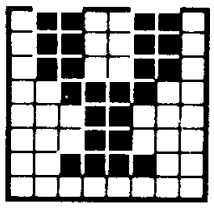
86
&H56
&X01010110



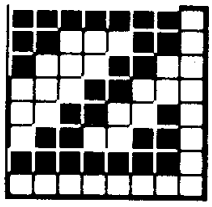
87
&H57
&X01010111



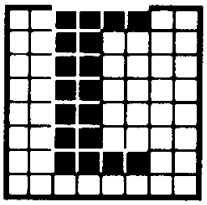
88
&H58
&X01011000



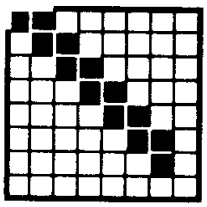
89
&H59
&X01011001



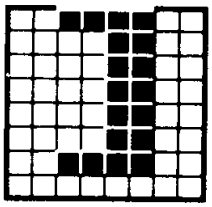
90
&H5A
&X01011010



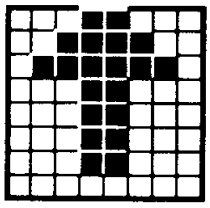
91
&H5B
&X01011011



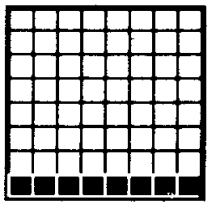
92
&H5C
&X01011100



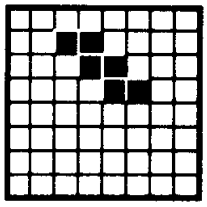
93
&H5D
&X01011101



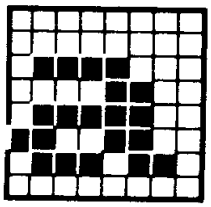
94
&H5E
&X01011110



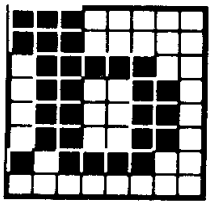
95
&H5F
&X01011111



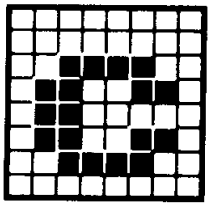
96
&H60
&X01100000



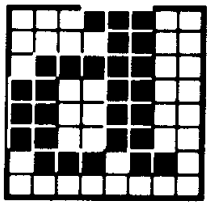
97
&H61
&X01100001



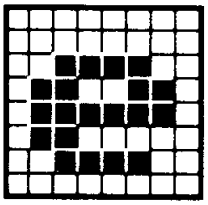
98
&H62
&X01100010



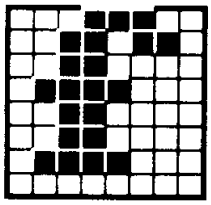
99
&H63
&X01100011



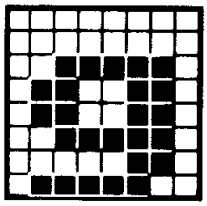
100
&H64
&X01100100



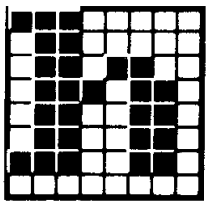
101
&H65
&X01100101



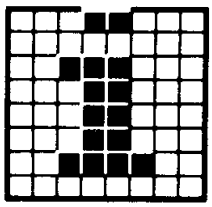
102
&H66
&X01100110



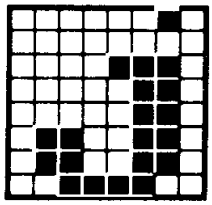
103
&H67
&X01100111



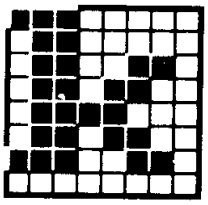
104
&H68
&X01101000



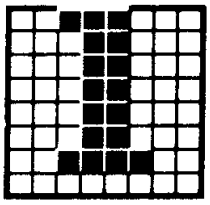
105
&H69
&X01101001



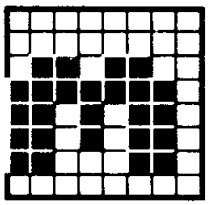
106
&H6A
&X01101010



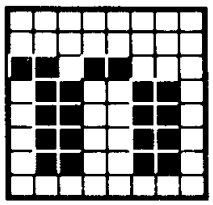
107
&H6B
&X01101011



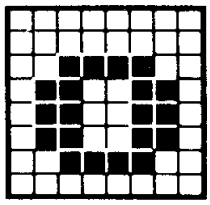
108
&H6C
&X01101100



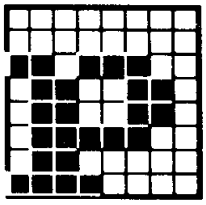
109
&H6D
&X01101101



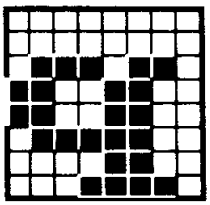
110
&H6E
&X01101110



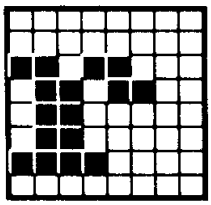
111
&H6F
&X01101111



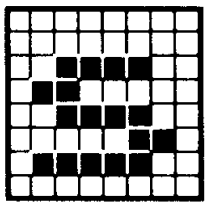
112
&H70
&X01110000



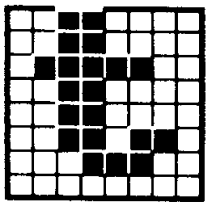
113
&H71
&X01110001



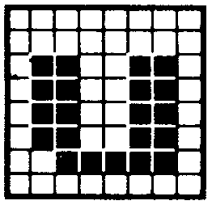
114
&H72
&X01110010



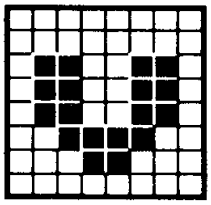
115
&H73
&X01110011



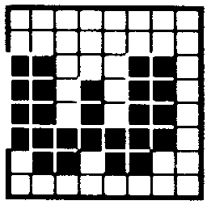
116
&H74
&X01110100



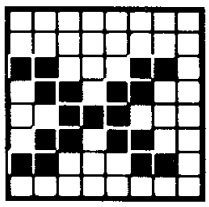
117
&H75
&X01110101



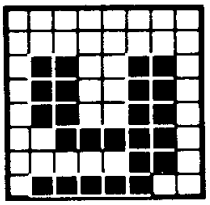
118
&H76
&X01110110



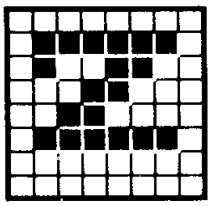
119
&H77
&X01110111



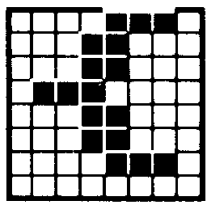
120
&H78
&X01111000



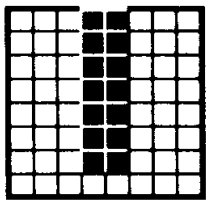
121
&H79
&X01111001



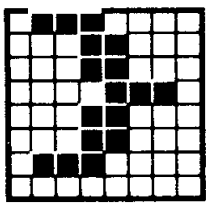
122
&H7A
&X01111010



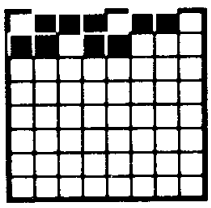
123
&H7B
&X01111011



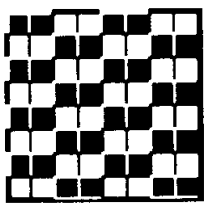
124
&H7C
&X01111100



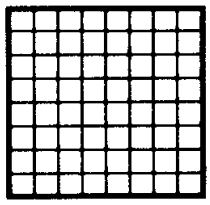
125
&H7D
&X01111101



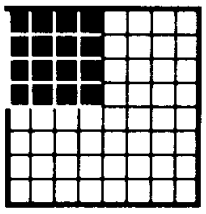
126
&H7E
&X01111110



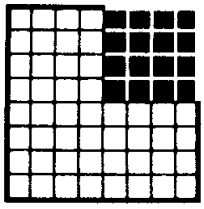
127
&H7F
&X01111111



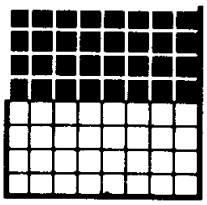
128
&H80
&X10000000



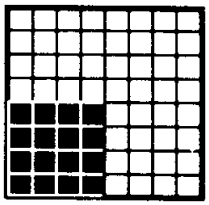
129
&H81
&X10000001



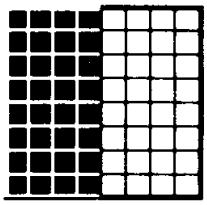
130
&H82
&X10000010



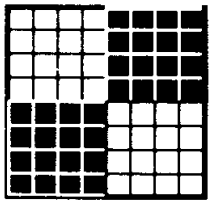
131
&H83
&X10000011



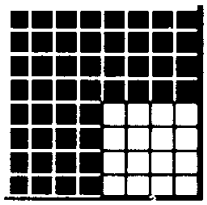
132
&H84
&X10000100



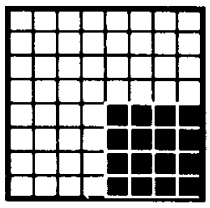
133
&H85
&X10000101



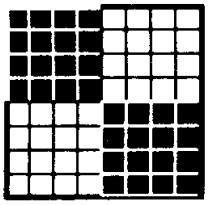
134
&H86
&X10000110



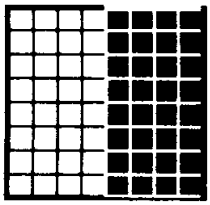
135
&H87
&X10000111



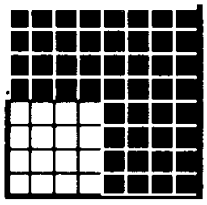
136
&H88
&X10001000



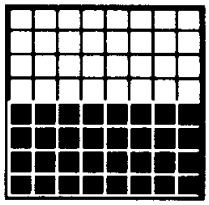
137
&H89
&X10001001



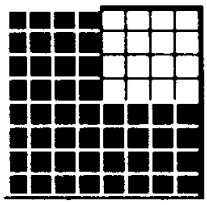
138
&H8A
&X10001010



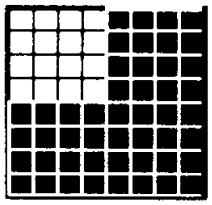
139
&H8B
&X10001011



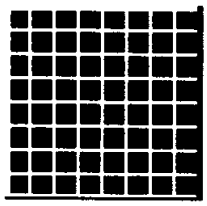
140
&H8C
&X10001100



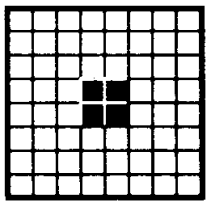
141
&H8D
&X10001101



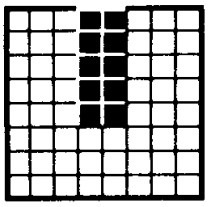
142
&H8E
&X10001110



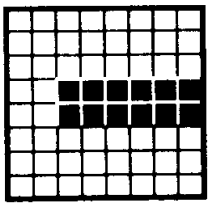
143
&H8F
&X10001111



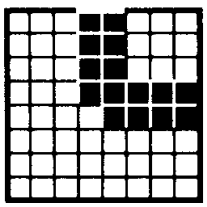
144
&H90
&X10010000



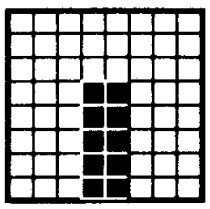
145
&H91
&X10010001



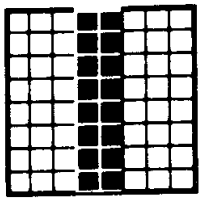
146
&H92
&X10010010



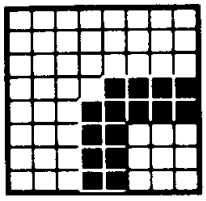
147
&H93
&X10010011



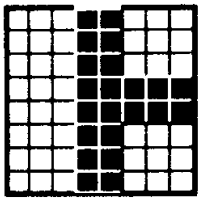
148
&H94
&X10010100



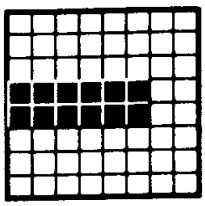
149
&H95
&X10010101



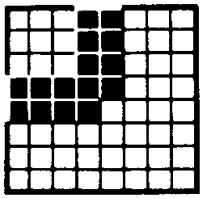
150
&H96
&X10010110



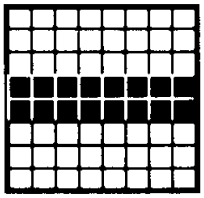
151
&H97
&X10010111



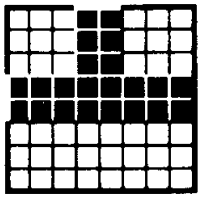
152
&H98
&X10011000



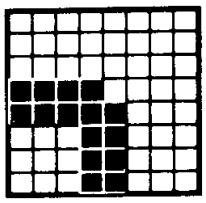
153
&H99
&X10011001



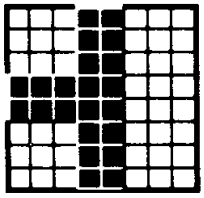
154
&H9A
&X10011010



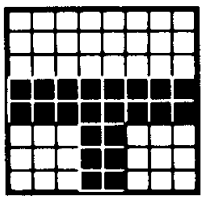
155
&H9B
&X10011011



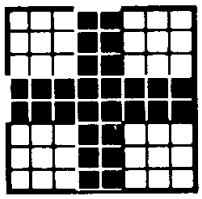
156
&H9C
&X10011100



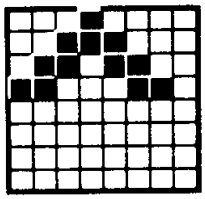
157
&H9D
&X10011101



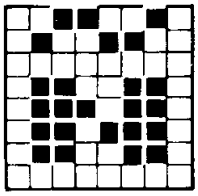
158
&H9E
&X10011110



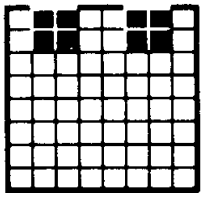
159
&H9F
&X10011111



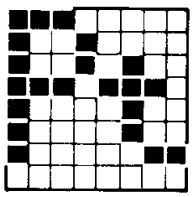
160
&HA0
&X10100000



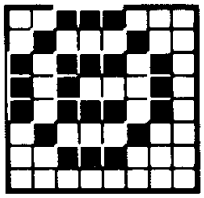
161
&HA1
&X10100001



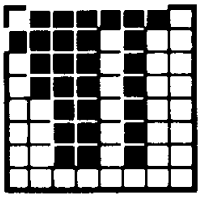
162
&HA2
&X10100010



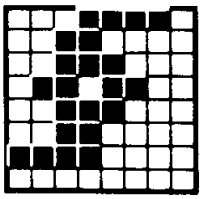
163
&HA3
&X10100011



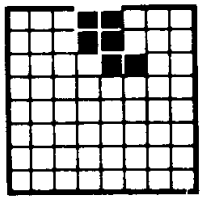
164
&HA4
&X10100100



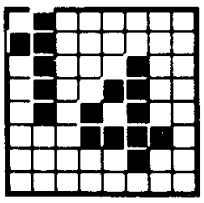
165
&HA5
&X10100101



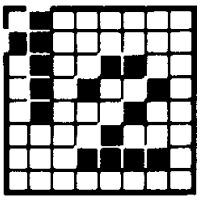
166
&HA6
&X10100110



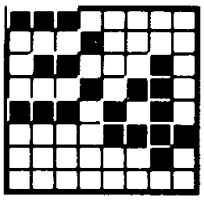
167
&HA7
&X10100111



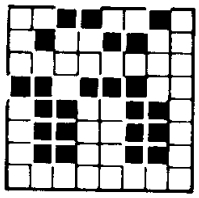
168
&HA8
&X10101000



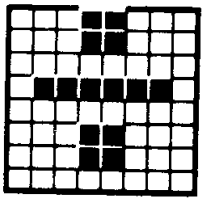
169
&HA9
&X10101001



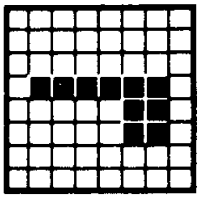
170
&HAA
&X10101010



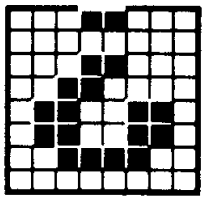
171
&HAB
&X10101011



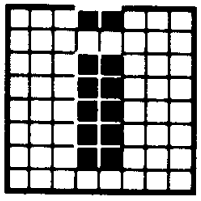
172
&HAC
&X10101100



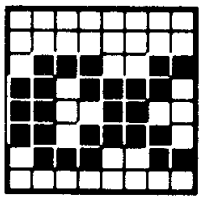
173
&HAD
&X10101101



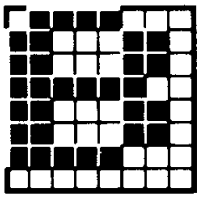
174
&HAE
&X10101110



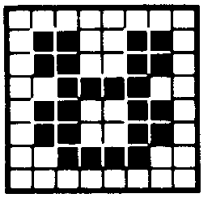
175
&HAF
&X10101111



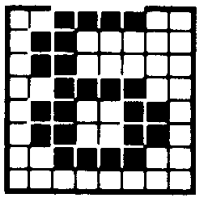
176
&HB0
&X10110000



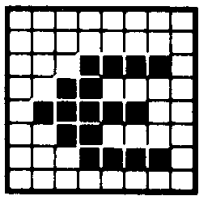
177
&HB1
&X10110001



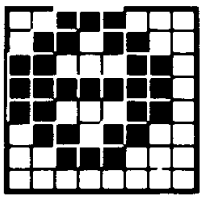
178
&HB2
&X10110010



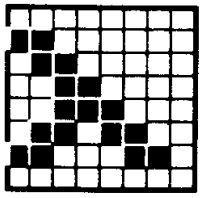
179
&HB3
&X10110011



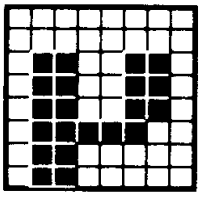
180
&HB4
&X10110100



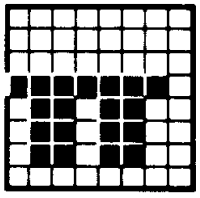
181
&HB5
&X10110101



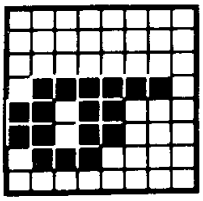
182
&HB6
&X10110110



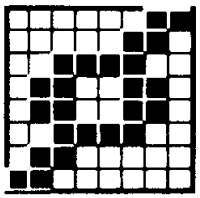
183
&HB7
&X10110111



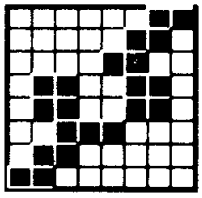
184
&HB8
&X10111000



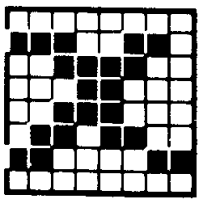
185
&HB9
&X10111001



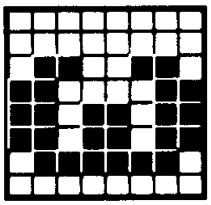
186
&HBA
&X10111010



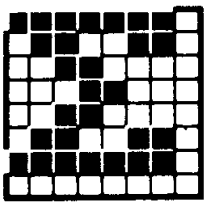
187
&HBB
&X10111011



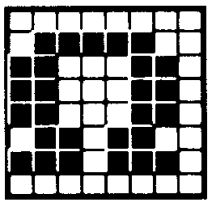
188
&HBC
&X10111100



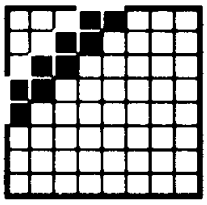
189
&HBD
&X10111101



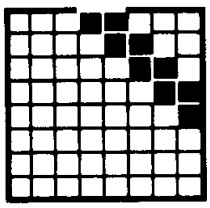
190
&HBE
&X10111110



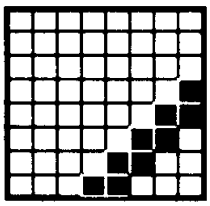
191
&HBF
&X10111111



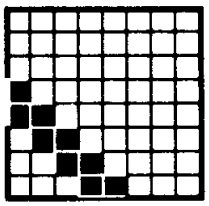
192
&HC0
&X11000000



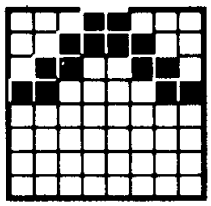
193
&HC1
&X11000001



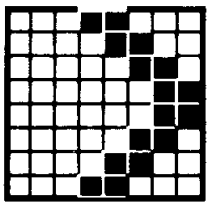
194
&HC2
&X11000010



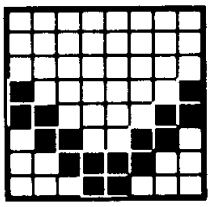
195
&HC3
&X11000011



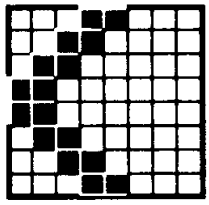
196
&HC4
&X11000100



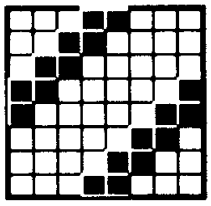
197
&HC5
&X11000101



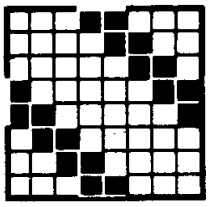
198
&HC6
&X11000110



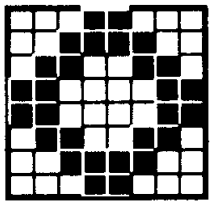
199
&HC7
&X11000111



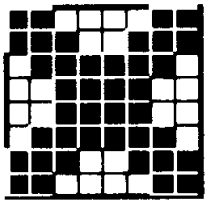
200
&HC8
&X11001000



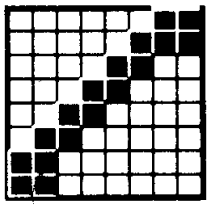
201
&HC9
&X11001001



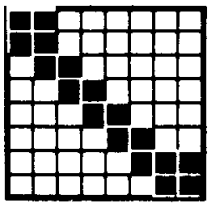
202
&HCA
&X11001010



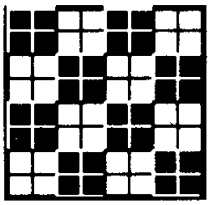
203
&HCB
&X11001011



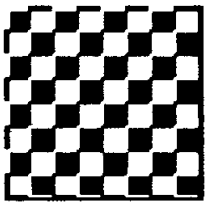
204
&HCC
&X11001100



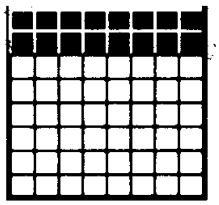
205
&HCD
&X11001101



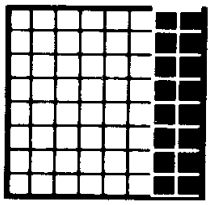
206
&HCE
&X11001110



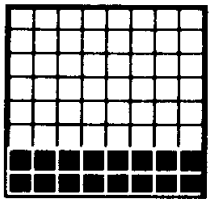
207
&HCF
&X11001111



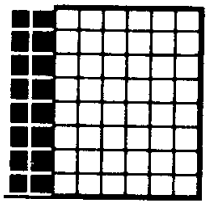
208
&HD0
&X11010000



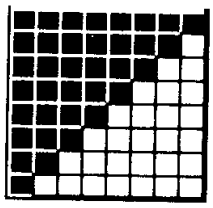
209
&HD1
&X11010001



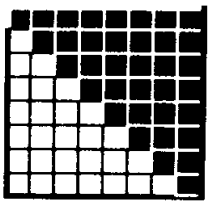
210
&HD2
&X11010010



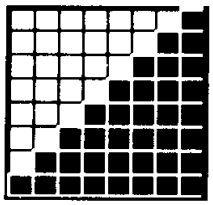
211
&HD3
&X11010011



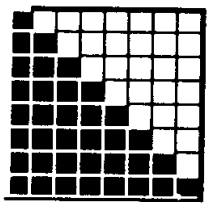
212
&HD4
&X11010100



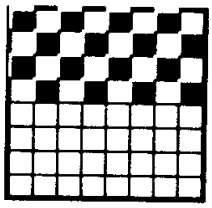
213
&HD5
&X11010101



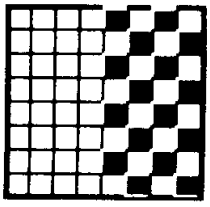
214
&HD6
&X11010110



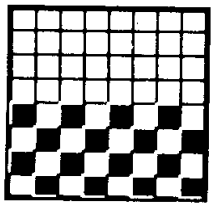
215
&HD7
&X11010111



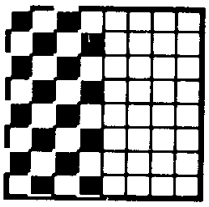
216
&HD8
&X11011000



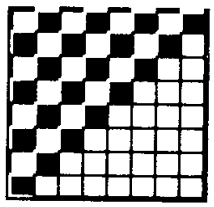
217
&HD9
&X11011001



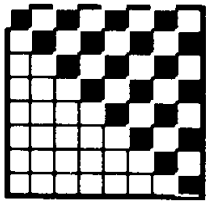
218
&HDA
&X11011010



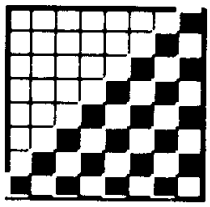
219
&HDB
&X11011011



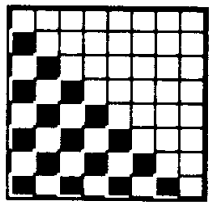
220
&HDC
&X11011100



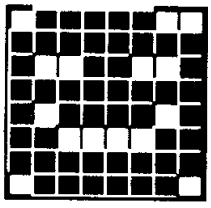
221
&HDD
&X11011101



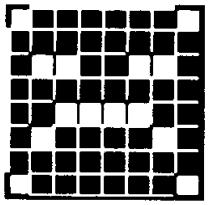
222
&HDE
&X11011110



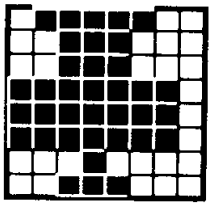
223
&HDF
&X11011111



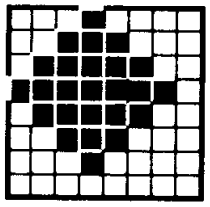
224
&HE0
&X11100000



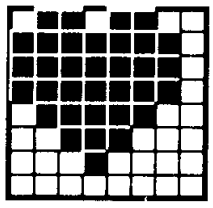
225
&HE1
&X11100001



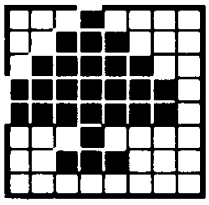
226
&HE2
&X11100010



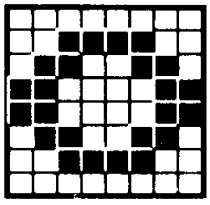
227
&HE3
&X11100011



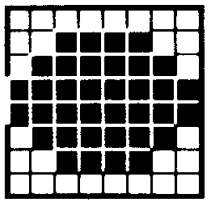
228
&HE4
&X11100100



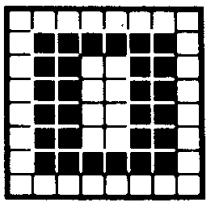
229
&HE5
&X11100101



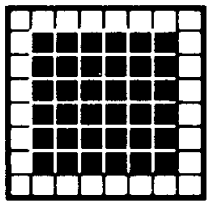
230
&HE6
&X11100110



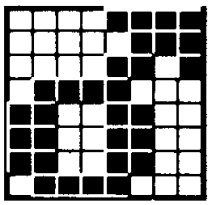
231
&HE7
&X11100111



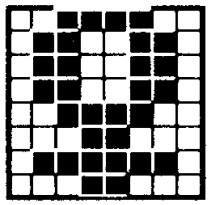
232
&HE8
&X11101000



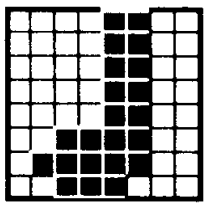
233
&HE9
&X11101001



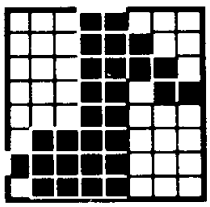
234
&HEA
&X11101010



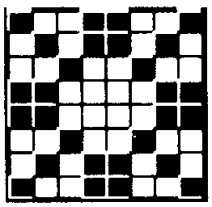
235
&HEB
&X11101011



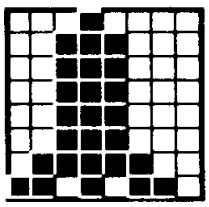
236
&HEC
&X11101100



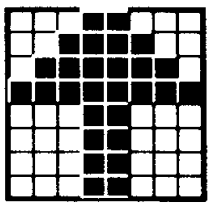
237
&HED
&X11101101



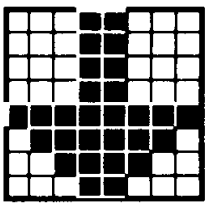
238
&HEE
&X11101110



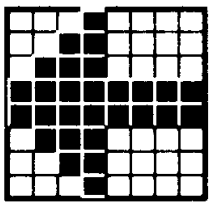
239
&HEF
&X11101111



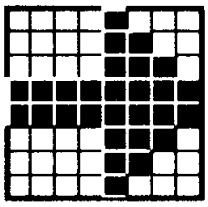
240
&HF0
&X11110000



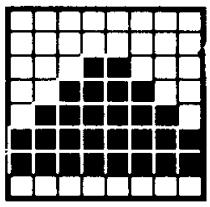
241
&HF1
&X11110001



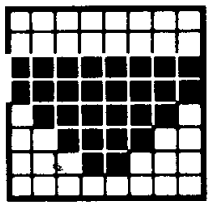
242
&HF2
&X11110010



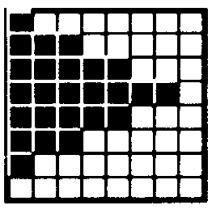
243
&HF3
&X11110011



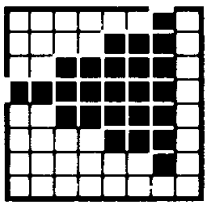
244
&HF4
&X11110100



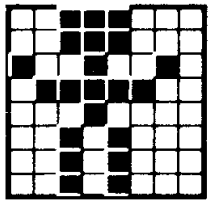
245
&HF5
&X11110101



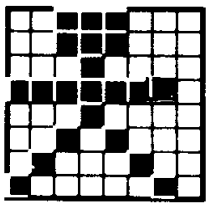
246
&HF6
&X11110110



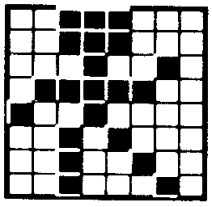
247
&HF7
&X11110111



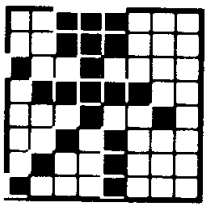
248
&HF8
&X11111000



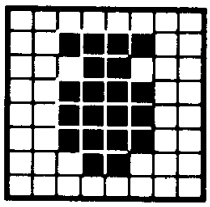
249
&HF9
&X11111001



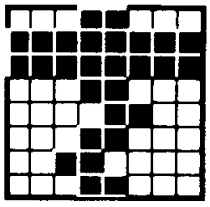
250
&HFA
&X11111010



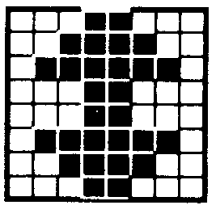
251
&HFB
&X11111011



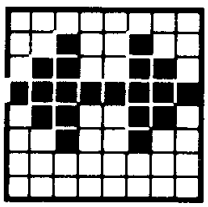
252
&HFC
&X11111100



253
&HFD
&X11111101



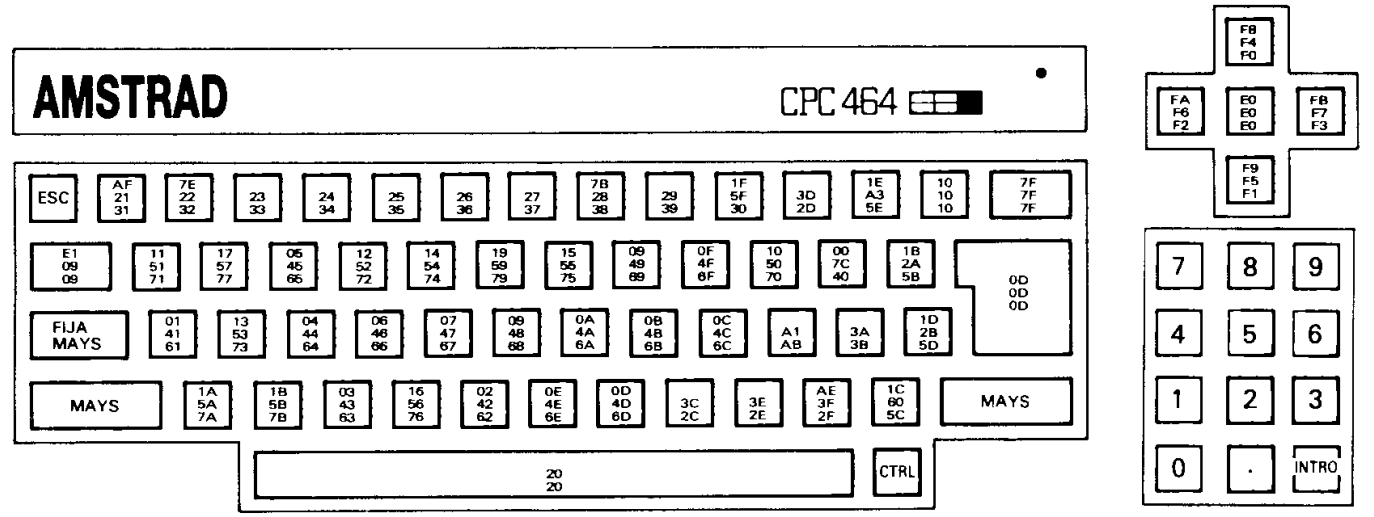
254
&HFE
&X11111110

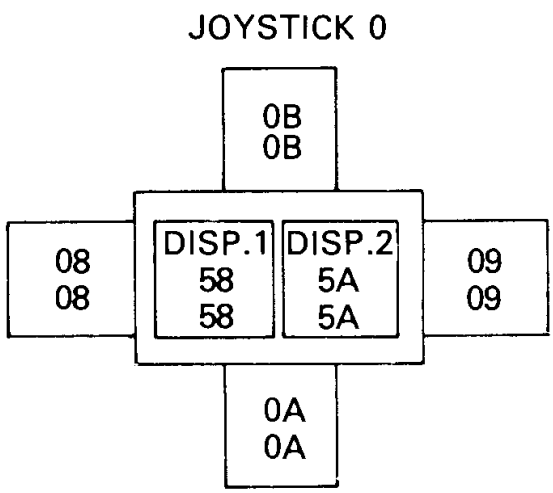


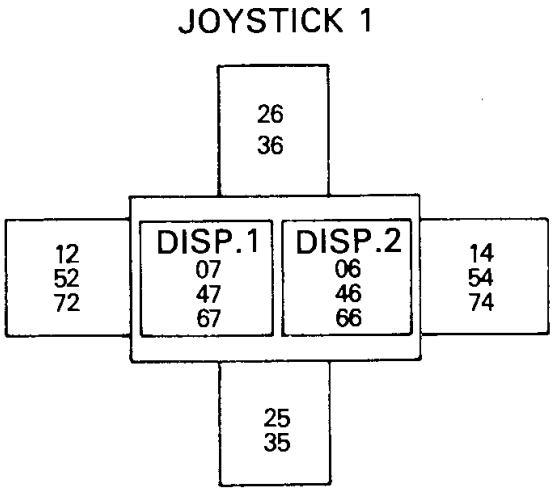
255
&HFF
&X11111111

Parte 5: Esquemas del teclado

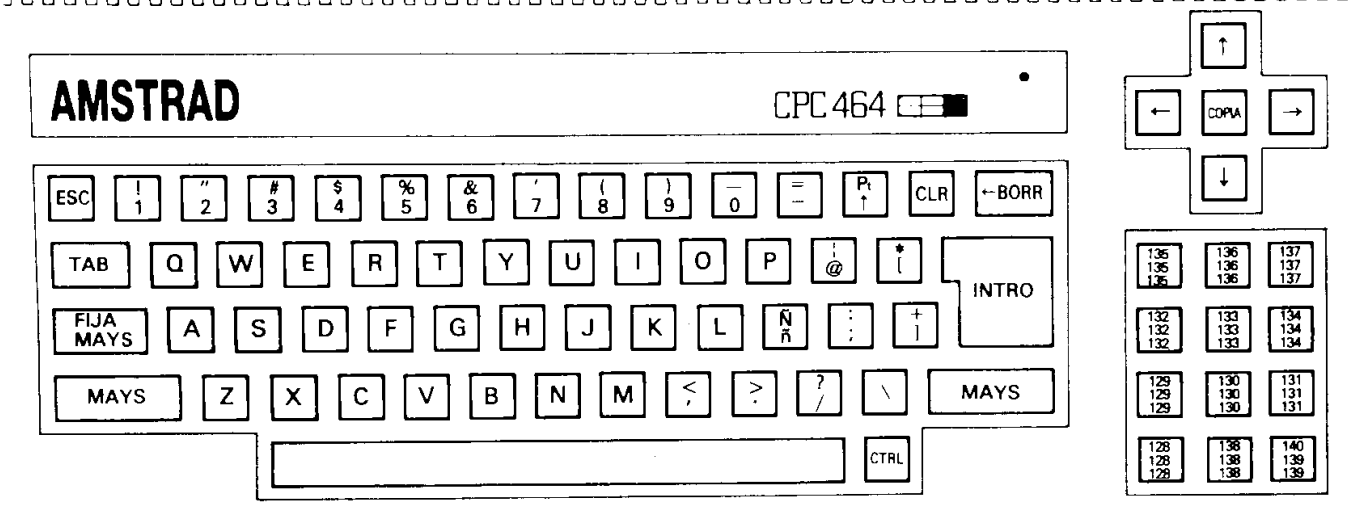
Valores ASCII implícitos generados por las teclas (HEX)







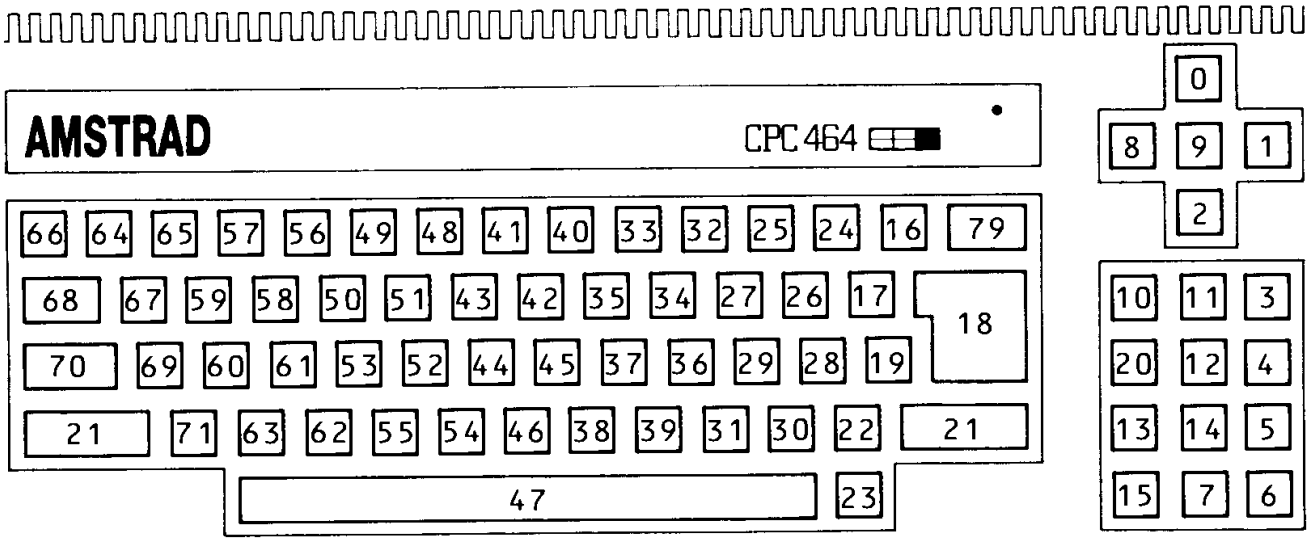
Códigos expansibles: posiciones y valores implícitos

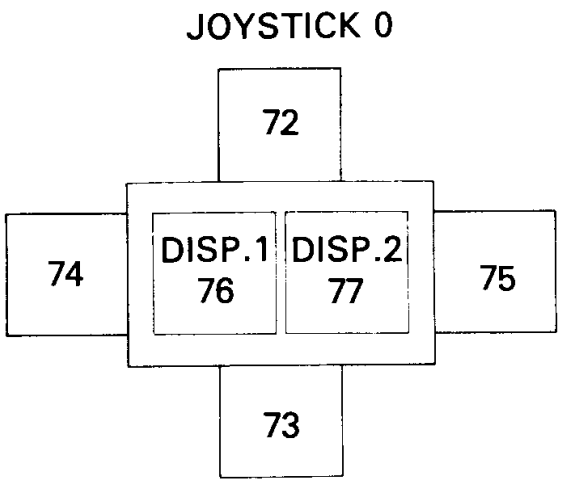


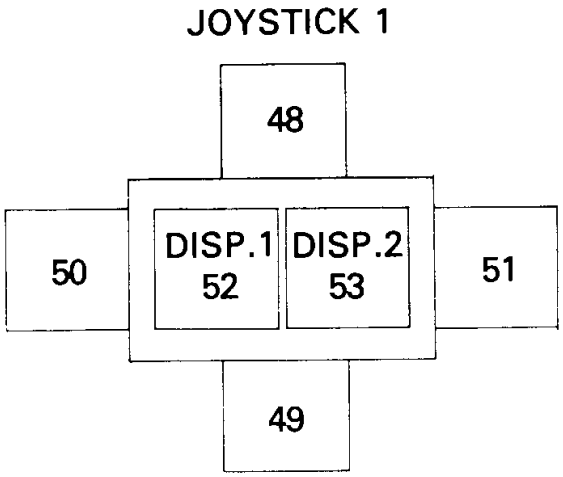
Código expansible	Asignación implícita	
	Carácter	Valor ASCII
0(128)	0	&30
1(129)	1	&31
2(130)	2	&32
3(131)	3	&33
4(132)	4	&34
5(133)	5	&35
6(134)	6	&36
7(135)	7	&37
8(136)	8	&38
9(137)	9	&39
10(138)	.	&2E
11(139)	INTRO	&0D
12(140)	RUN" INTRO	&52&55&4E&22&0D

Nota. Los caracteres expansibles 13 a 31 (141 a 159) no tienen asignada ninguna cadena de expansión. Su expansión se puede definir con la orden **KEY** de BASIC, y se los puede asignar a teclas concretas con la orden **KEY DEF**.

Números de las teclas y los joysticks







Parte 6: Sonidos

Notas y periodos de tono

La tabla siguiente da los valores recomendados para el parámetro ⟨periodo de tono⟩ correspondientes a los tonos y semitonos de las ocho octavas que puede producir el CPC464.

Las frecuencias generadas no coinciden exactamente con las requeridas porque el ⟨periodo de tono⟩ tiene que ser un número entero. El 'error relativo' es el cociente:

$$\frac{\text{frecuencia requerida} - \text{frecuencia generada}}{\text{frecuencia requerida}}$$

expresado en porcentaje.

NOTA		FRECUENCIA	PERIODO	ERROR RELATIVO	
C	DO	32.703	3822	-0.007%	
C#	DO#	34.648	3608	+0.007%	
D	RE	36.708	3405	-0.007%	
D#	RE#	38.891	3214	-0.004%	
E	MI	41.203	3034	+0.009%	
F	FA	43.654	2863	-0.016%	
F#	FA#	46.249	2703	+0.009%	Octava -3
G	SOL	48.999	2551	-0.002%	
G#	SOL#	51.913	2408	+0.005%	
A	LA	55.000	2273	+0.012%	
A#	LA#	58.270	2145	-0.008%	
B	SI	61.735	2025	+0.011%	
NOTA		**FRECUENCIA**	**PERIODO**	**ERROR RELATIVO**	
C	DO	65.406	1911	-0.007%	
C#	DO#	69.296	1804	+0.007%	
D	RE	73.416	1703	+0.022%	
D#	RE#	77.782	1607	-0.004%	
E	MI	82.407	1517	+0.009%	
F	FA	87.307	1432	+0.019%	
F#	FA#	92.499	1351	-0.028%	Octava -2
G	SOL	97.999	1276	+0.037%	
G#	SOL#	103.826	1204	+0.005%	
A	LA	110.000	1136	-0.032%	
A#	LA#	116.541	1073	+0.039%	
B	SI	123.471	1012	-0.038%	

NOTA		FRECUENCIA	PERIODO	ERROR RELATIVO	
C	DO	130.813	956	+0.046%	
C#	DO#	138.591	902	+0.007%	
D	RE	146.832	851	-0.037%	
D#	RE#	155.564	804	+0.058%	
E	MI	164.814	758	-0.057%	
F	FA	174.614	716	+0.019%	
F#	FA#	184.997	676	+0.046%	Octava -1
G	SOL	195.998	638	+0.037%	
G#	SOL#	207.652	602	+0.005%	
A	LA	220.000	568	-0.032%	
A#	LA#	233.082	536	-0.055%	
B	SI	246.942	506	-0.038%	

NOTA		FRECUENCIA	PERIODO	ERROR RELATIVO	
C	DO	261.626	478	+0.046%	DO media
C#	DO#	277.183	451	+0.007%	
D	RE	293.665	426	+0.081%	
D#	RE#	311.127	402	+0.058%	
E	MI	329.628	379	-0.057%	
F	FA	349.228	358	+0.019%	
F#	FA#	369.994	338	+0.046%	Octava 0
G	SOL	391.995	319	+0.037%	
G#	SOL#	415.305	301	+0.005%	
A	LA	440.000	284	-0.032%	LA internacional
A#	LA#	466.164	268	-0.055%	
B	SI	493.883	253	-0.038%	

NOTA		FRECUENCIA	PERIODO	ERROR RELATIVO	
C	DO	523.251	239	+0.046%	
C#	DO#	554.365	225	-0.215%	
D	RE	587.330	213	+0.081%	
D#	RE#	622.254	201	+0.058%	
E	MI	659.255	190	+0.206%	
F	FA	698.457	179	+0.019%	
F#	FA#	739.989	169	+0.046%	Octava 1
G	SOL	783.991	159	-0.277%	
G#	SOL#	830.609	150	-0.328%	
A	LA	880.000	142	-0.032%	
A#	LA#	932.328	134	-0.055%	
B	SI	987.767	127	+0.356%	

NOTA		FRECUENCIA	PERIODO	ERROR RELATIVO	
C	DO	1046.502	119	-0.374%	
C#	DO#	1108.731	113	+0.229%	
D	RE	1174.659	106	-0.390%	
D#	RE#	1244.508	100	-0.441%	
E	MI	1318.510	95	+0.206%	
F	FA	1396.913	89	-0.543%	
F#	FA#	1479.978	84	-0.548%	Octava 2
G	SOL	1567.982	80	+0.350%	
G#	SOL#	1661.219	75	-0.328%	
A	LA	1760.000	71	-0.032%	
A#	LA#	1864.655	67	-0.055%	
B	SI	1975.533	63	-0.435%	

NOTA		FRECUENCIA	PERIODO	ERROR RELATIVO	
C	DO	2093.004	60	+0.462%	
C#	DO#	2217.461	56	-0.662%	
D	RE	2349.318	53	-0.390%	
D#	RE#	2489.016	50	-0.441%	
E	MI	2637.021	47	-0.855%	
F	FA	2793.826	45	+0.574%	
F#	FA#	2959.955	42	-0.548%	Octava 3
G	SOL	3135.963	40	+0.450%	
G#	SOL#	3322.438	38	+0.992%	
A	LA	3520.000	36	+1.357%	
A#	LA#	3729.310	34	+1.417%	
B	SI	3951.066	32	+1.134%	

NOTA		FRECUENCIA	PERIODO	ERROR RELATIVO	
C	DO	4186.009	30	+0.462%	
C#	DO#	4434.922	28	-0.662%	
D	RE	4698.636	27	+1.469%	
D#	RE#	4978.032	25	-0.441%	
E	MI	5274.041	24	+1.246%	
F	FA	5587.652	22	-1.685%	
F#	FA#	5919.911	21	-0.548%	Octava 4
G	SOL	6271.927	20	+0.350%	
G#	SOL#	6644.875	19	+0.992%	
A	LA	7040.000	18	+1.357%	
A#	LA#	7458.621	17	+1.417%	
B	SI	7902.133	16	+1.134%	

Estas frecuencias han sido calculadas a partir de la correspondiente a la nota LA internacional mediante la siguiente fórmula:

```
FRECUENCIA=440*(2↑(OCTAVA+((N-10)/12)))
PERIODO=ROUND(125000/FRECUENCIA)
```

donde **N** es 1 para DO, 2 para DO#, etc.

Parte 7: Mensajes de error de BASIC

1 Unexpected NEXT

Se ha encontrado un **NEXT** que no corresponde a ningún **FOR**, o bien la variable mencionada tras **NEXT** no coincide con la de **FOR**.

2 Syntax error

BASIC no puede entender la línea porque algo en ella no se ajusta a las reglas de sintaxis del lenguaje.

3 Unexpected RETURN

Se ha encontrado una orden **RETURN** fuera de una subrutina.

4 DATA exhausted

Una orden **READ** ha intentado leer más datos de los incluidos en líneas **DATA**.

5 Improper argument

Este es un error no específico. Puede haber algo incorrecto en el argumento de una función o en un parámetro de una orden.

6 Overflow

El resultado de una operación aritmética ha sobrepasado los márgenes aceptables. Si ha ocurrido en un número en punto flotante, el resultado ha sido mayor que 1.7E+38 (aprox.). También puede haber ocurrido al intentar convertir un número en punto flotante en un entero con signo de 16 bits.

7 Memory full

El programa actual o sus variables ocupan demasido espacio en la memoria, el anidamiento de la estructura de control es demasiado profundo (demasiadas órdenes **GOSUB**, **WHILE** o **FOR** anidadas).

La orden **MEMORY** provoca este error si se intenta situar el límite de la memoria accesible a BASIC demasiado bajo, o tan alto que sea imposible. Obsérvese que los ficheros abiertos tienen un tampón dedicado a ellos, lo que puede restringir los valores correctos de **MEMORY**.

8 Line does not exist

Se ha mencionado un número de línea que no existe en el programa.

9 Subscript out of range

Uno de los subíndices mencionados en una lista es demasiado pequeño o demasiado grande.

10 Array already dimensioned

Una de las listas declaradas en una orden **DIM** ya ha sido dimensionada antes.

11 Division by zero

División por cero, que puede ocurrir en la división de números reales, en la división entera, en la operación **MOD** y en la exponenciación.

12 Invalid direct command

Se ha intentado dar como orden directa una orden que sólo es válida si va precedida de un número de línea.

13 Type mismatch

Incongruencia de tipos. Un dato es literal cuando tenía que ser numérico, o viceversa, o se ha introducido un dato no numérico en **READ** o **INPUT**.

14 String space full

Se han creado demasiadas cadenas literales y ya no hay espacio disponible para más, incluso después de reorganizar la memoria.

15 String too long

Una cadena literal consta de más de 255 caracteres. Puede haber sido generada por concatenación de otras.

16 String expression too complex

Expresión literal demasiado compleja. Las expresiones literales pueden generar valores intermedios; cuando el número de estos valores excede de un límite razonable, se produce este error.

17 Cannot CONTinue

No se puede reanudar la ejecución del programa con **CONT**. Obsérvese que **CONT** reanuda el programa después de una instrucción **STOP**, de [ESC] [ESC] o de un error, pero a condición de que no se modifique el programa durante la pausa.

18 Unknown user function

Se ha invocado una **FN** sin la correspondiente **DEF FN**.

19 RESUME missing

Se ha llegado al final del programa mientras se ejecutaba una rutina de gestión de errores (esto es, una rutina invocada por **ON ERROR GOTO**).

20 Unexpected RESUME

RESUME sólo es válida en una rutina de gestión de errores (esto es, en una rutina invocada por **ON ERROR GOTO**).

21 Direct command found

Al cargar un fichero se ha encontrado una línea sin número de línea.

22 Operand missing

BASIC ha encontrado una expresión incompleta.

23 Line too long

BASIC ha convertido a su forma interna una línea y ésta ha resultado ser demasiado larga.

24 EOF met

Se ha intentado leer más allá del final del fichero de entrada.

25 File type error

El fichero que se ha leído no es del tipo adecuado. **OPENIN** sólo puede abrir ficheros de texto ASCII. Análogamente, **LOAD**, **RUN**, etc., solamente pueden acceder a ficheros creados por **SAVE**.

26 NEXT missing

BASIC ha encontrado un **FOR** al que no corresponde ningún **NEXT**. La línea mencionada tras el mensaje es aquella en la que se encuentra el **FOR**.

27 File already open

Se ha ejecutado una orden **OPENIN** u **OPENOUT** sin previamente cerrar el fichero que estaba abierto.

28 Unknown command

BASIC no encuentra un comando externo que corresponda a lo que se ha escrito tras |.

29 WEND missing

BASIC ha encontrado un **WHILE** al que no corresponde ningún **WEND**.

30 Unexpected WEND

Se ha encontrado un **WEND** que no corresponde a ningún **WHILE**, o bien el **WEND** no concuerda con el **WHILE** del bucle actual.

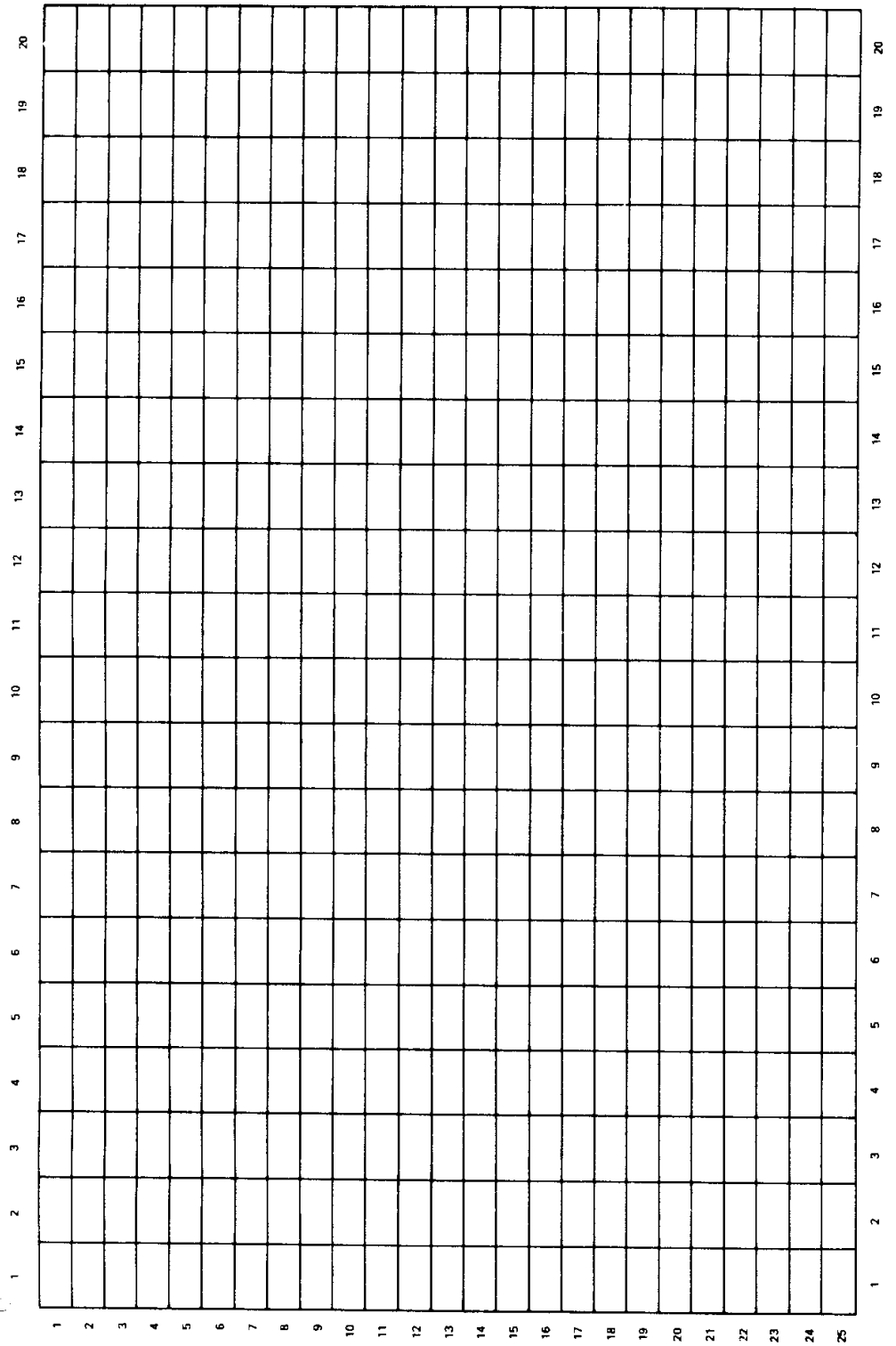
Parte 8: Palabras reservadas de BASIC

A continuación damos la lista de las palabras reservadas para el BASIC de AMSTRAD, las cuales, por consiguiente, **no** pueden ser utilizadas como nombres de variables.

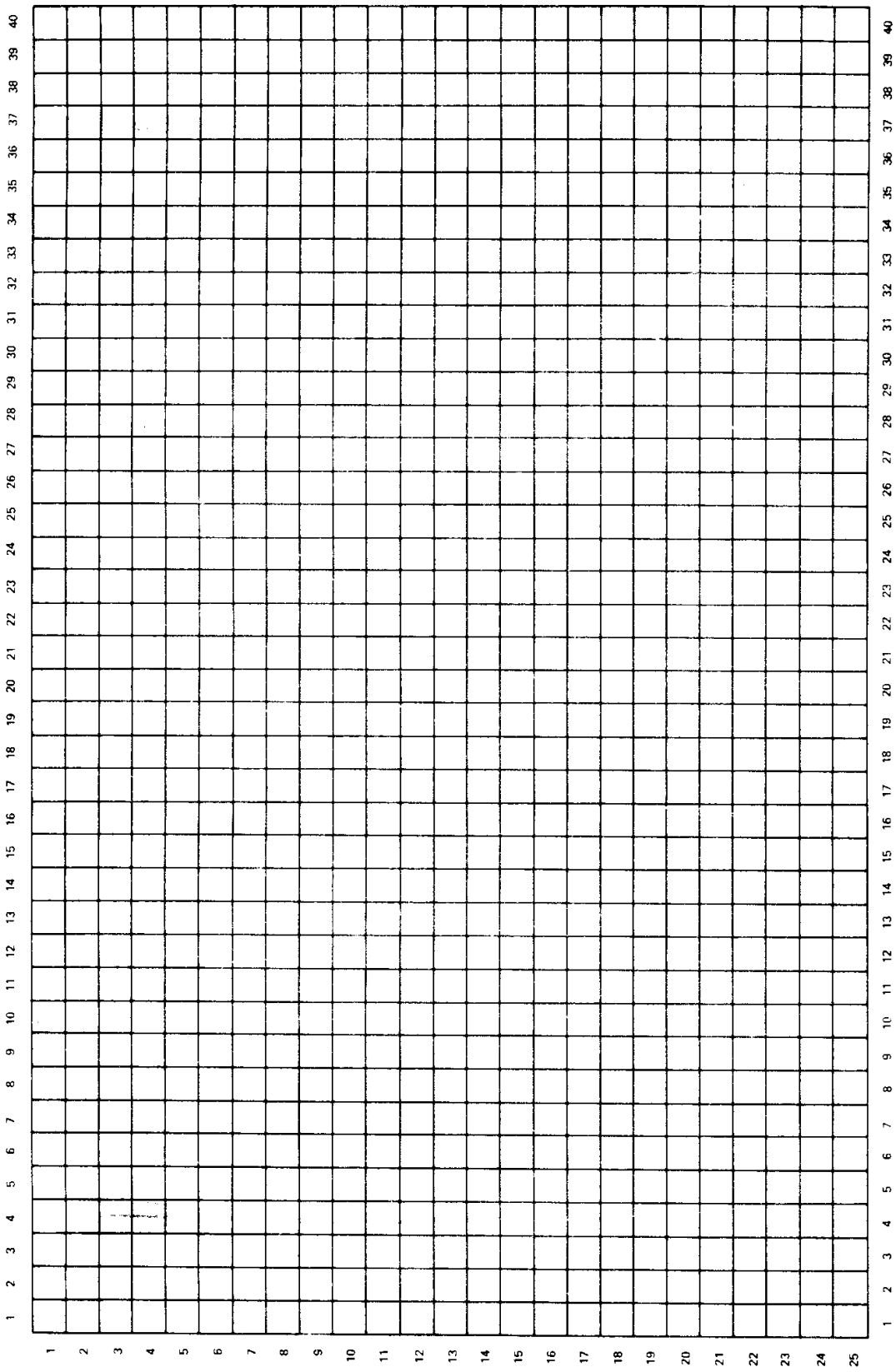
ABS, AFTER, AND, ASC, ATN, AUTO
BIN$, BORDER
CALL, CAT, CHAIN, CHR$, CINT, CLEAR, CLG, CLOSEIN, CLOSEOUT, CLS, CONT, COS, CREAL
DATA, DEF, DEFINT, DEFREAL, DEFSTR, DEG, DELETE, DI, DIM, DRAW, DRAWR
EDIT, EI, ELSE, END, ENT, ENV, EOF, ERASE, ERL, ERR, ERROR, EVERY, EXP
FIX, FN, FOR, FRE
GOSUB, GOTO
HEX$, HIMEM
IF, INK, INKEY, INKEY$, INP, INPUT, INSTR, INT
JOY
KEY
LEFT$, LEN, LET, LINE, LIST, LOAD, LOCATE, LOG, LOG10, LOWER$
MAX, MEMORY, MERGE, MID$, MIN, MOD, MODE, MOVE, MOVER
NEXT, NEW, NOT
ON, ON BREAK, ON ERROR GOTO 0, ON SQ, OPENIN, OPENOUT, OR, ORIGIN, OUT
PAPER, PEEK, PEN, PI, PLOT, PLOTR, POKE, POS, PRINT
RAD, RANDOMIZE, READ, RELEASE, REM, REMAIN, RENUM, RESTORE, RESUME, RETURN, RIGHT$, RND, ROUND, RUN
SAVE, SGN, SIN, SOUND, SPACE$, SPC, SPEED, SQ, SQR, STEP, STOP, STR$, STRING$, SWAP, SYMBOL
TAB, TAG, TAGOFF, TAN, TEST, TESTR, THEN, TIME, TO, TROFF, TRON
UNT, UPPER$, USING
VAL, VPOS
WAIT, WEND, WHILE, WIDTH, WINDOW, WRITE
XOR, XPOS
YPOS
ZONE

Parte 9: Plantillas

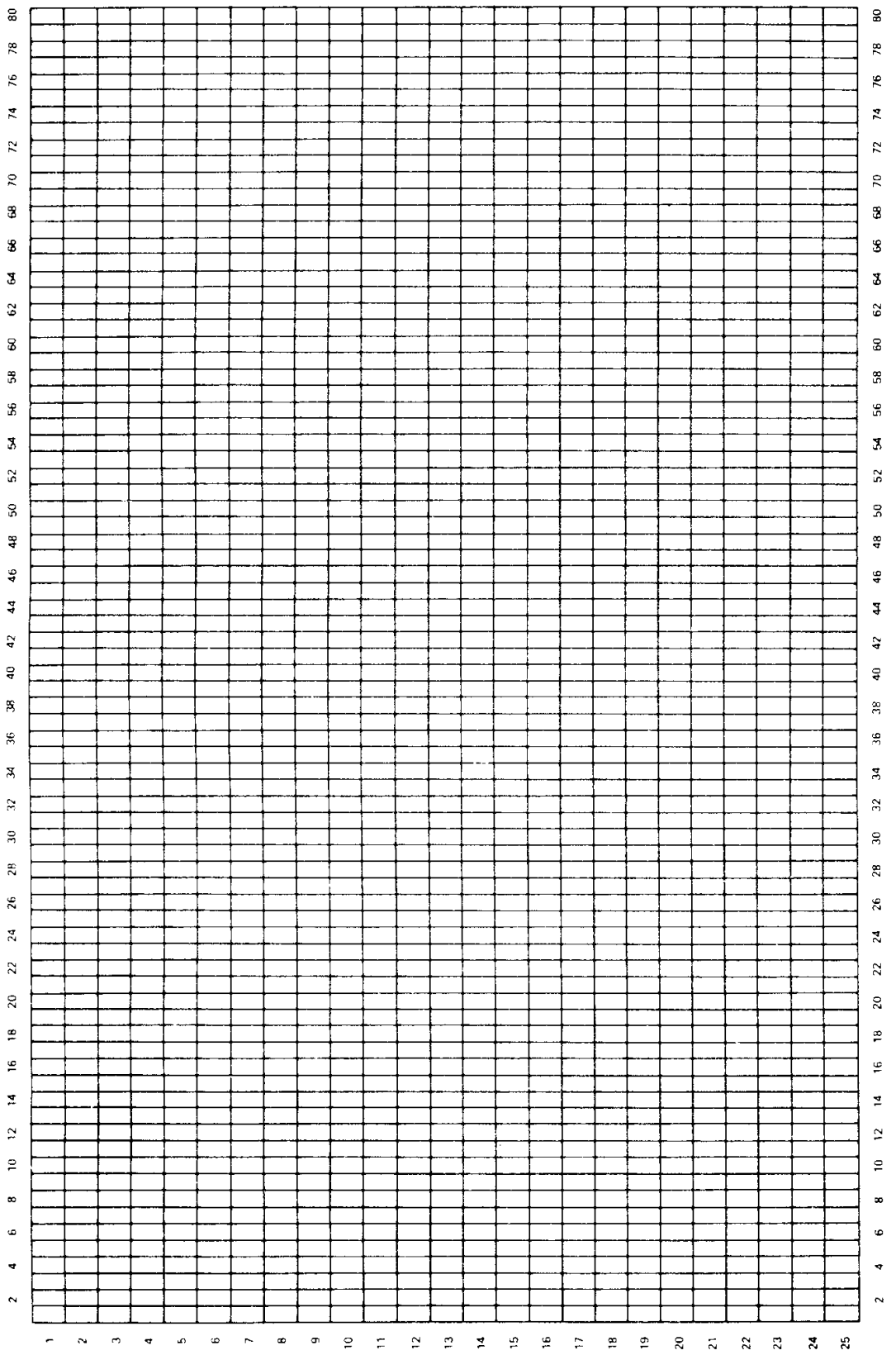
Pantalla de texto; modo 0 (20 columnas)



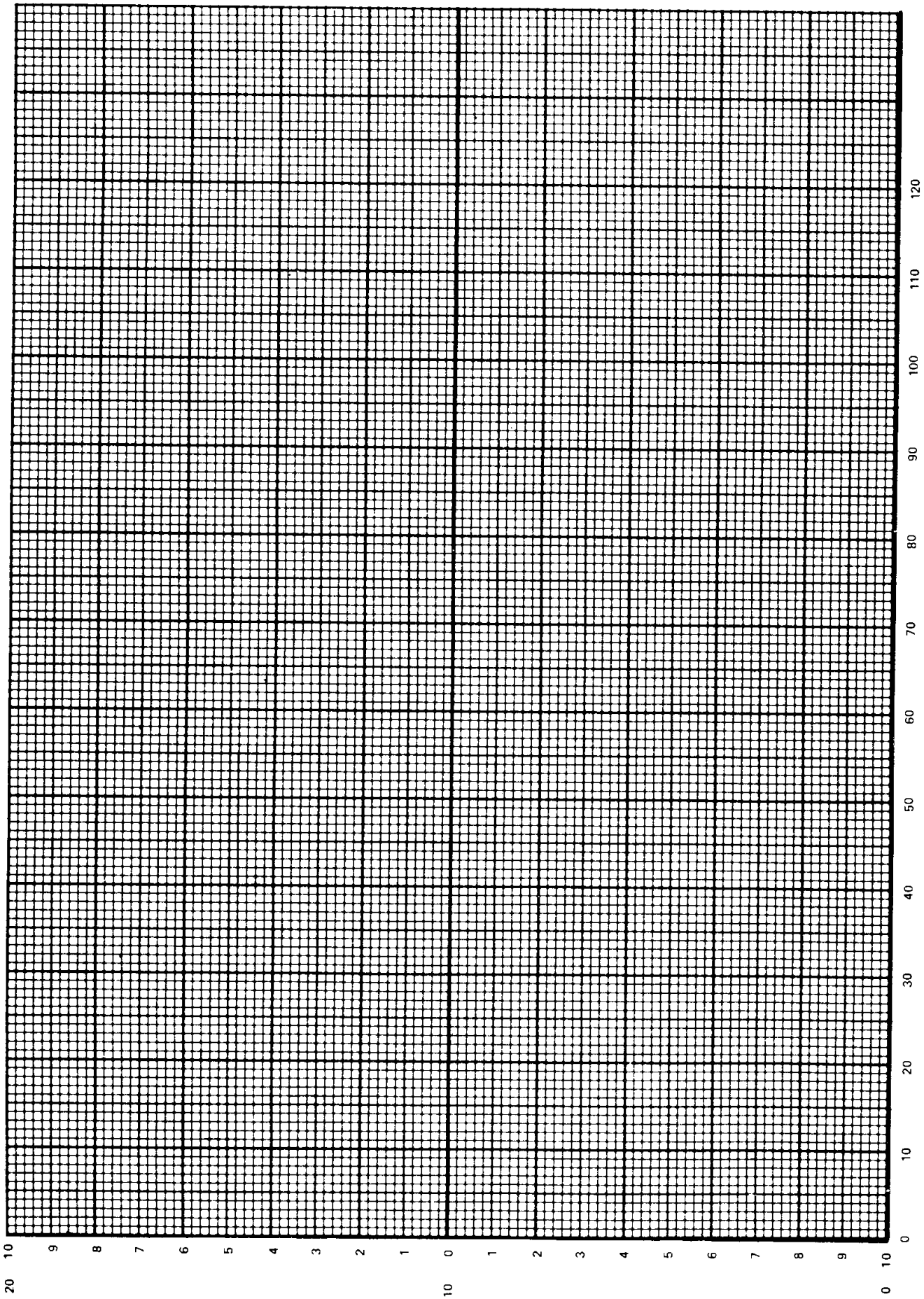
Pantalla de texto; modo 1 (40 columnas)



Pantalla de texto; modo 2 (80 columnas)

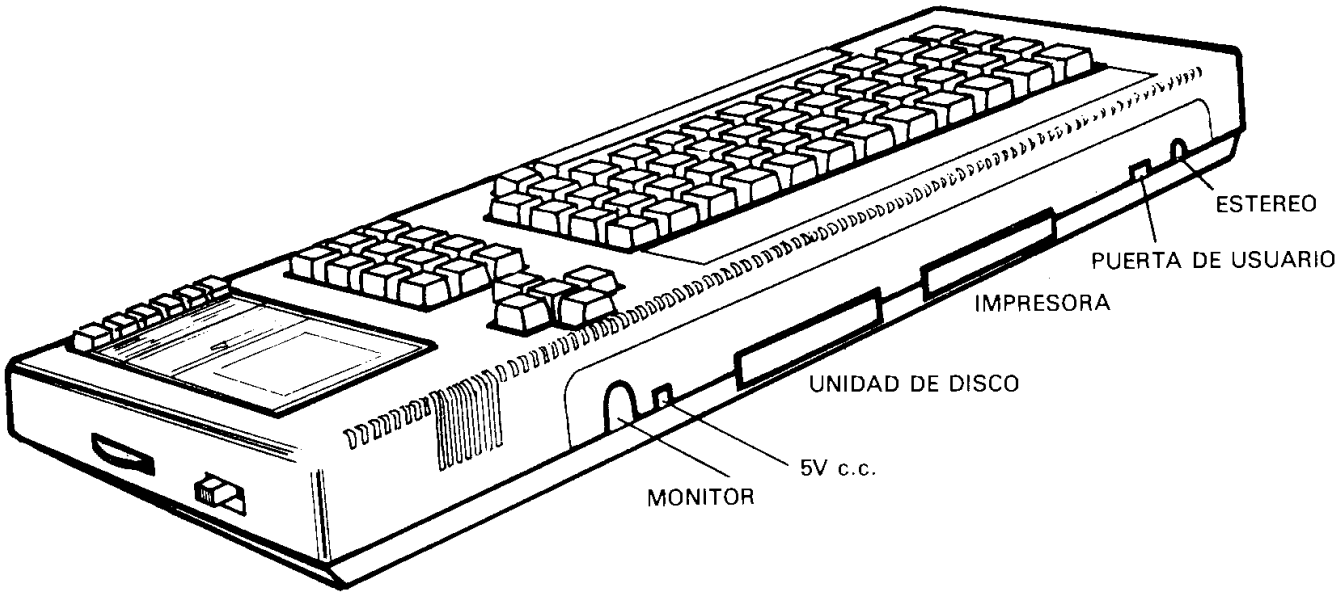


Envolventes de sonido/música



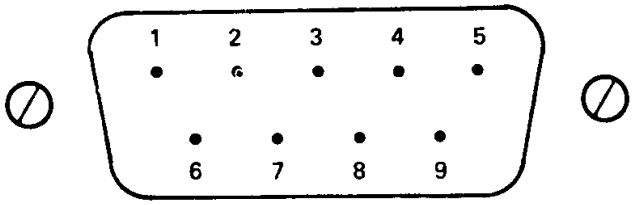
Parte 10: Conexiones

Zócalos de entradas y salidas del CPC464



Zócalo del joystick

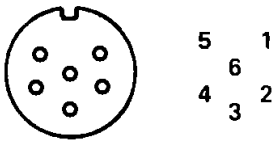
VISTO POR DETRAS



Patilla 1 ARRIBA	Patilla 6 DISPARO 2
Patilla 2 ABAJO	Patilla 7 DISPARO 1
Patilla 3 IZQUIERDA	Patilla 8 MASA 2
Patilla 4 DERECHA	Patilla 9 MASA
Patilla 5 N/C	

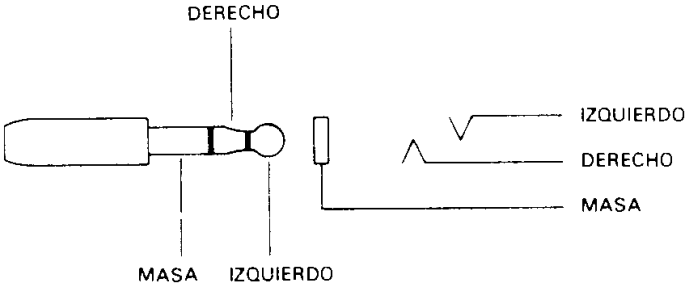
Zócalo del monitor

VISTO POR DETRAS



Patilla 1 ROJO	Patilla 4 SINC
Patilla 2 VERDE	Patilla 5 MASA
Patilla 3 AZUL	Patilla 6 LUM

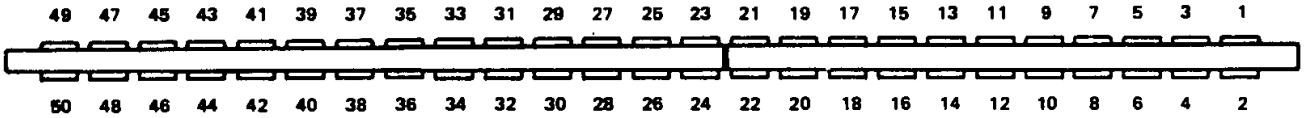
Zócalo estéreo



Patilla 1 CANAL IZQUIERDO
Patilla 2 CANAL DERECHO
Patilla 3 MASA

Zócalo de unidad de disco

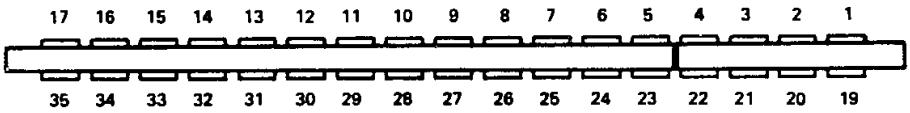
VISTO POR DETRAS



Patilla 1 SONIDO	Patilla 18 A0	Patilla 35 $\overline{\text{INT}}$
Patilla 2 MASA	Patilla 19 D7	Patilla 36 $\overline{\text{NMI}}$
Patilla 3 A15	Patilla 20 D6	Patilla 37 $\overline{\text{BUSR2}}$
Patilla 4 A14	Patilla 21 D5	Patilla 38 $\overline{\text{BUSAK}}$
Patilla 5 A13	Patilla 22 D4	Patilla 39 READY
Patilla 6 A12	Patilla 23 D3	Patilla 40 $\overline{\text{BUS RESET}}$
Patilla 7 A11	Patilla 24 D2	Patilla 41 $\overline{\text{RESET}}$
Patilla 8 A10	Patilla 25 D1	Patilla 42 $\overline{\text{ROMEN}}$
Patilla 9 A9	Patilla 26 D0	Patilla 43 ROMDIS
Patilla 10 A8	Patilla 27 +5v	Patilla 44 $\overline{\text{RAMRD}}$
Patilla 11 A7	Patilla 28 $\overline{\text{MREQ}}$	Patilla 45 RAMDIS
Patilla 12 A6	Patilla 29 $\overline{\text{MI}}$	Patilla 46 CURSOR
Patilla 13 A5	Patilla 30 $\overline{\text{RFSH}}$	Patilla 47 LAPIZ ÓPTICO
Patilla 14 A4	Patilla 31 $\overline{\text{IORQ}}$	Patilla 48 $\overline{\text{EXP}}$
Patilla 15 A3	Patilla 32 $\overline{\text{RD}}$	Patilla 49 MASA
Patilla 16 A2	Patilla 33 $\overline{\text{WR}}$	Patilla 50
Patilla 17 A1	Patilla 34 $\overline{\text{HALT}}$	

Zócalo para la impresora

VISTO POR DETRAS



Patilla 1 $\overline{\text{STROBE}}$	Patilla 19 MASA
Patilla 2 D0	Patilla 20 MASA
Patilla 3 D1	Patilla 21 MASA
Patilla 4 D2	Patilla 22 MASA
Patilla 5 D3	Patilla 23 MASA
Patilla 6 D4	Patilla 24 MASA
Patilla 7 D5	Patilla 25 MASA
Patilla 8 D6	Patilla 26 MASA
Patilla 9 MASA	Patilla 27 MASA
Patilla 11 BUSY	Patilla 28 MASA
Patilla 14 MASA	Patilla 33 MASA
Patilla 16 MASA	Todas las demás: N/C

Parte 11: Impresoras

Interfaz

Al CPC464 se le puede conectar cualquier impresora que tenga 'interfaz paralelo tipo Centronics' de 7 bits.

El cable es fácil de construir, pues no tiene que realizar más que una conexión biunívoca entre el zócalo **IMPRESORA** del ordenador y el conector de la impresora. Nótese que en la salida del ordenador hay dos terminales menos que en el conector de la impresora, con objeto de permitir la utilización de un conector estándar para circuito impreso.

Los detalles de las patillas del interfaz están ilustrados en la parte 9 de este capítulo.

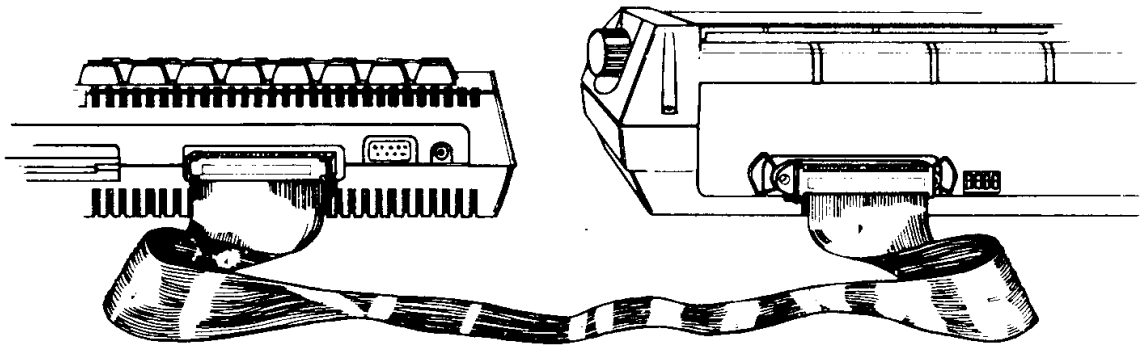
El cable debe conectar el terminal 1 del ordenador con el 1 de la impresora, el 19 del ordenador con el 19 de la impresora, etc. Sin embargo, los terminales 18 y 36 de la impresora *no* deben ser conectados al ordenador. Obsérvese que aunque los terminales de la fila superior del zócalo **IMPRESORA** del ordenador están numerados del 1 al 17, la numeración de la fila inferior empieza en el 19, no en el 18, para mantener la correspondencia con el conector de la impresora.

El ordenador utiliza la señal BUSY (patilla 11) para sincronizar con la impresora y espera cuando ésta está en fuera de línea.

No es necesario enviar a la impresora códigos de preparación; la salida del ordenador se dirige a la impresora especificando el canal #**8**.

La salida **IMPRESORA** está diseñada para conectar impresoras de matriz de puntos de bajo costo. No obstante, con el interfaz adecuado, se le pueden conectar impresoras de margarita, plotters e impresoras de color por chorro de tinta. La compatibilidad está asegurada por el interfaz paralelo.

La programación incorporada a la impresora AMSTRAD DMP1 facilita el funcionamiento en modo gráfico así como los volcados de pantalla.



Parte 12: Joysticks

El ordenador puede funcionar con uno o dos joysticks. La máquina los considera como parte del teclado, y por consiguiente su estado se puede determinar con **INKEY** e **INKEY$**.

Nótese que en la mayor parte de los casos, el botón principal de disparo del joystick es interpretado como 'DISP.2' por el CPC464.

Las funciones **JOY(0)** y **JOY(1)** permiten inspeccionar directamente los dos joysticks. Cada una de estas funciones genera un resultado, interpretable bit a bit, que informa de la situación de los interruptores del joystick en el momento en que se realizó la última comprobación del teclado.

La tabla siguiente indica los valores generados por los dos joysticks. Además de los números de la función **JOY**, se incluyen los números que se deben utilizar en instrucciones que llevan como parámetro los números de las teclas (tales como **INKEY** y **KEY DEF**).

Estado	Función **JOY**		Números de tecla		
	Bit que está a 1	Valor generado	Primer joystick	Segundo joystick	Tecla equivalente
Arriba	0	1	72	48	'6'
Abajo	1	2	73	49	'5'
Izquierda	2	4	74	50	'R'
Derecha	3	8	75	51	'T'
Disparo 2	4	16	76	52	'G'
Disparo 1	5	32	77	53	'F'

Nótese que cuando se generan los valores correspondientes al segundo joystick, el ordenador no puede saber si esos valores han sido generados por el joystick o por las teclas equivalentes (última columna de la tabla anterior). Esto representa que se puede utilizar el teclado como sustituto del segundo joystick.

Parte 13: Ampliaciones del sistema residentes (RSX)

El repertorio de instrucciones de BASIC puede ser ampliado añadiéndole 'órdenes externas'. En ocasiones, estas órdenes estarán grabadas en ROMs externas (por ejemplo, en la ROM de la unidad de disco opcional DDI-1, en la ROM del interfaz serie RS232C, etc.). Pero también es posible incorporar nuevas órdenes externas (cuando BASIC ya está activado) cargando instrucciones de código de máquina adecuadas en la RAM. Tales órdenes externas son las llamadas 'ampliaciones del sistema residentes' (RSX) y actúan exactamente igual que las basadas en ROM. Las RSX han de ser cargadas desde la cinta cada vez que se entra en BASIC (en frío o en caliente).

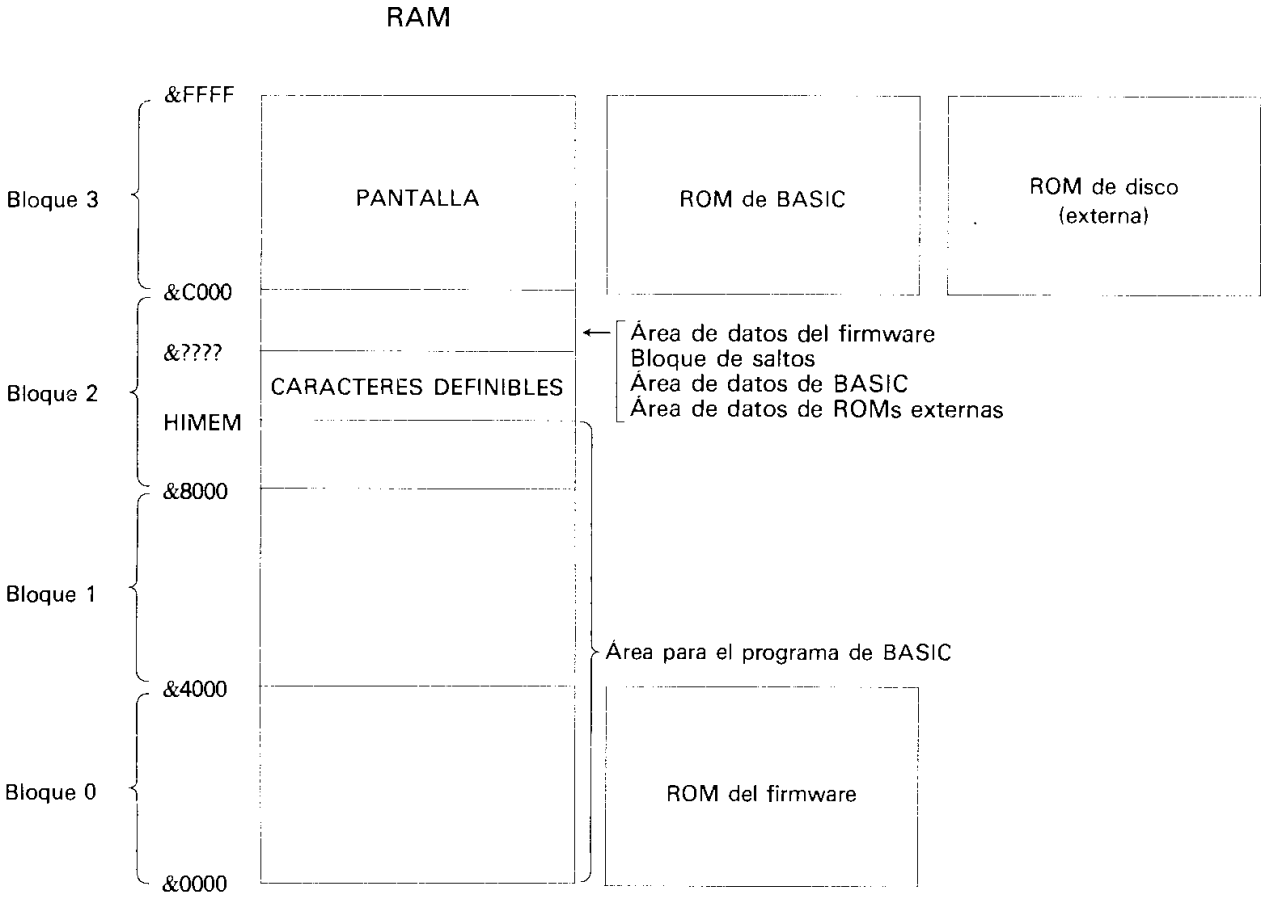
Todas las órdenes externas se escriben precedidas del signo |.

Entre sus aplicaciones típicas se encuentra el control de periféricos (unidad de disco, interfaz serie, lapicero fotosensible, . . .) y la ampliación de BASIC con nuevas funciones y rutinas (por ejemplo, para rellenar recintos, dibujar circunferencias, realizar volcados de pantalla en la impresora, etc.).

La información necesaria para diseñar y gestionar las RSX se puede encontrar en la 'Guía del firmware', SOFT 158.

Parte 14: Memoria

El CPC464 dispone de 64K de RAM y 32K de ROM. Esta memoria es utilizable por BASIC según se indica en la figura siguiente. La RAM está dividida en cuatro bloques de 16K cada uno, numerados del 0 al 3. La pantalla utiliza el bloque 3; la zona superior del bloque 2 está dedicada a las variables del sistema, según se muestra en la figura.



Nota. &???? depende de las ROMs externas conectadas.

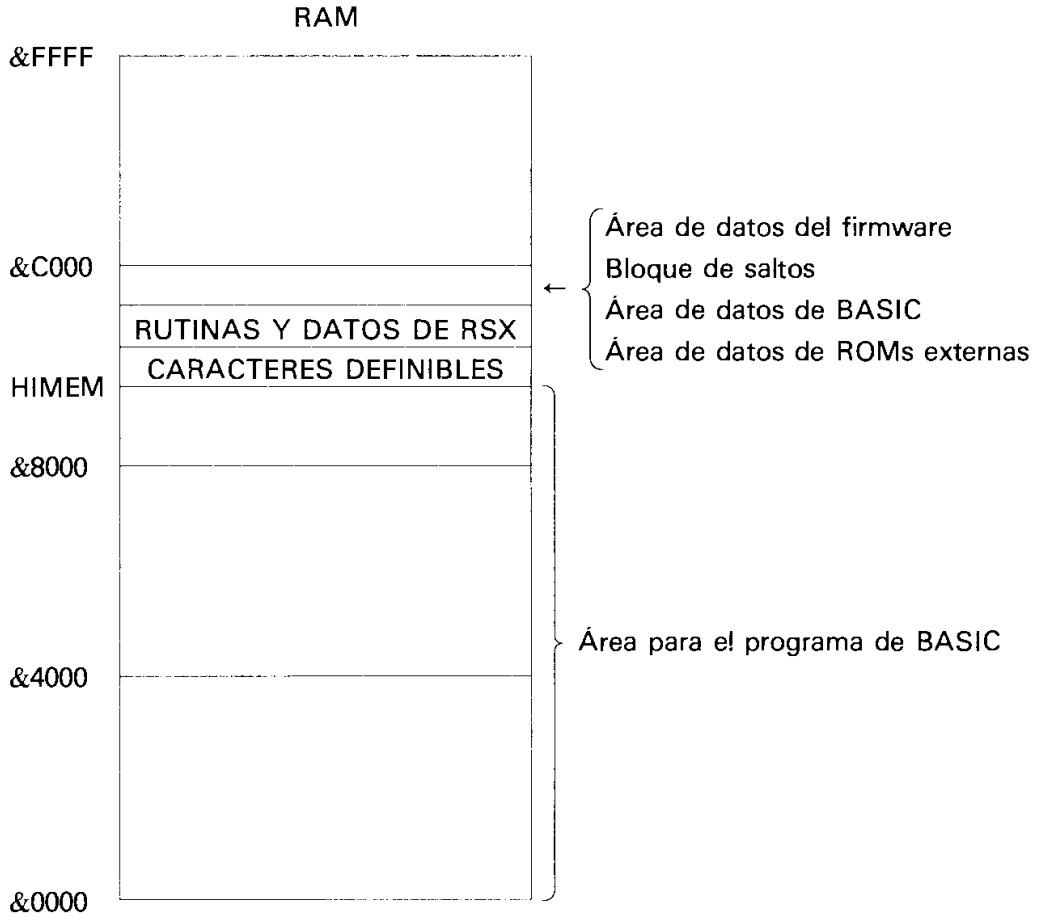
Los caracteres definidos por el usuario están inicialmente inmediatamente por encima de HIMEM. Este valor se puede modificar mediante una orden **MEMORY**; además se reduce en 4K automáticamente para crear un tampón cada vez que se abre un fichero. El número de caracteres definibles por el usuario (controlado por la orden **SYMBOL AFTER** de BASIC) sólo se puede modificar si HIMEM no ha variado desde la última vez que aquél fue establecido (o si en esa ocasión se estableció en cero, con **SYMBOL AFTER 256**). Cuando se entra en BASIC, el número de caracteres definibles por el usuario se establece como si se hubiera ejecutado la orden **SYMBOL AFTER 240**.

Por consiguiente, antes de cambiar permanentemente HIMEM, es prudente reducir a cero la zona de memoria dedicada a los caracteres definibles; después se modifica HIMEM y fi-

nalmente se restablece la zona de caracteres definibles al tamaño deseado. De esta menera los siguientes programas podrán ejecutar nuevas órdenes **SYMBOL AFTER** sin problemas.

El siguiente ejemplo ilustra este método, aplicado a la carga de una RSX:

```
100 SYMBOL AFTER 256 ' anular memoria para caracteres definibles por
    el usuario
110 direccionrxs=HIMEM-longitudrsx
120 MEMORY direccionrsx-1
130 LOAD "codigrsx",direccionrsx
140 CALL direccionrsx ' inicializar rsx
150 SYMBOL AFTER 140 ' restablecer espacio para caracteres definibles
```



Mapa de RAM con las RSX cargadas en la posición recomendada.

Otras entradas y salidas

La mayor parte de las direcciones de puertas de entrada/salida (E/S) está reservada por el ordenador. En particular, las direcciones inferiores a &7FFF no se deben utilizar nunca.

Está previsto que la parte de la dirección representada por A0-A7 refleje el tipo de dispositivo externo de E/S, y que las líneas de dirección A8 y A9 puedan ser decodificadas para seleccionar registros dentro del dispositivo de E/S. De las restantes líneas, sólo A10 puede ser decodificada (a nivel 'bajo'); las líneas A11 a A15 están todas a nivel 'alto'. Así, los registros de los dispositivos se pueden direccionar como &F8??, &F9??, &FA?? y &FB??, donde ?? puede estar entre DC y FF para los interfaces de comunicación y entre E0 y FE para otros periféricos.

Nótese que se deben utilizar instrucciones del Z80 que pongan el registro B en la mitad superior del bus de direcciones (A15-A8).

Memorias ROM adicionales

El sistema operativo está preparado para seleccionar ROMs adicionales en lugar de parte de la ROM interna. La lógica de gestión de direcciones y de selección de bancos estará en un módulo conectado al bus de expansión, pero todas las señales necesarias están en el corrector de expansión.

Capítulo 5
Cuando usted guste ...

En este capítulo vamos a dar un repaso a ciertas cuestiones básicas de informática en general y del CPC464 en particular. No es imprescindible leer este capítulo para trabajar con el ordenador, pero puede ayudarle a entender algo de lo que está ocurriendo en su interior.

Parte 1: En términos generales ...

¡Fulmina el bicho!

Aun cuando lo único que le interese de su CPC464 sea disfrutar de los sofisticados juegos de ordenador que se venden en las tiendas, es posible que se pregunte qué tiene la máquina dentro, es decir, qué es eso del *hardware*.

Hardware es lo que usted puede coger y llevarse bajo el brazo (es un decir): la caja gris donde está el teclado, el monitor, los cables, etc. Más en general, todo lo que no sea específicamente *software*: programas, manuales e información grabada en cinta o en disco.

Ciertos aspectos del funcionamiento del ordenador están estrechamente ligados al hardware; por ejemplo, la imagen de color que usted puede ver en la pantalla. En cambio, es responsabilidad del software controlar el hardware para producir los caracteres y dibujos deseados.

Es el hardware el que dirige el haz de electrones a la superficie electroluminiscente de que está recubierto el interior del tubo del televisor; el software aporta la organización y la inteligencia para decir al hardware cómo y cuándo debe hacerlo. Añade además el control de los tiempos y de las secuencias de acciones para que usted pueda ver la nave espacial despegando, o para algo tan mundano como que aparezca una letra en la pantalla cada vez que usted pulsa una tecla.

¿Por qué es un ordenador mejor que otro?

El hardware sin el software es completamente inútil, y viceversa. El ordenador empieza a servir para algo cuando hardware y software se ponen de acuerdo para llevar a cabo sus cometidos. Hay algunas consideraciones básicas que se pueden tener en cuenta para valorar la calidad tanto de uno como del otro.

Los puntos de referencia generalmente aceptados para los ordenadores personales son:

1. Resolución de la pantalla: tamaño del menor elemento discernible

El concepto de resolución es una mezcla de diversos factores: número de colores disponibles para el programador, número de áreas discernibles en la pantalla (pixels) y número de caracteres de texto que caben en ella.

Usted puede comprobar que su CPC464 es superior a este respecto a los otros ordenadores de su nivel de precio.

2. El intérprete de BASIC

Prácticamente todos los ordenadores domésticos incluyen un intérprete de BASIC que permite al usuario crear programas con los que controlar el hardware. El lenguaje de programación que viene con la máquina, BASIC, no es más que un programa, pero un programa extremadamente complejo que ha evolucionado a lo largo de los años, desde que se 'inventó' el BASIC en los Estados Unidos. El 'Beginners' All-purpose Symbolic Instruction Code' (código de instrucciones simbólicas de uso general para principiantes) es el lenguaje de ordenador más ampliamente utilizado en el mundo entero; como todos los lenguajes, tiene sus dialectos locales.

La versión utilizada en el CPC464 es una de las de mayor compatibilidad; puede ejecutar programas escritos para funcionar bajo control del sistema operativo de disco CP/M. Es una versión muy rápida de BASIC, lo que quiere decir que realiza los cálculos muy deprisa. A usted quizá no le parezca demasiado impresionante que un ordenador tarde 0.05 segundos en multiplicar 3 por 5 y escribir la respuesta, mientras que otro ordenador tarda 0.075 segundos en hacer lo mismo. Pero cuando se trata de dibujar gráficos en la pantalla, tarea que puede requerir millares de cálculos repetitivos, la diferencia entre 0.05 y 0.075 segundos es decisiva.

Usted oirá hablar frecuentemente de 'códigos de máquina'. El código de máquina es una forma de suministrar instrucciones directamente al microprocesador. La velocidad de ejecución es entre 5 y 15 veces mayor en código de máquina que en BASIC, pero el tiempo que se tarda en escribir un programa puede ser hasta 50 veces mayor en código de máquina que en BASIC.

El BASIC de Amstrad es de los más rápidos y completos que se pueden encontrar en ordenadores personales, e incorpora muchos recursos que ayudan al programador experto a superar parte de la lentitud inherente a todos los lenguajes de alto nivel para producir efectos visuales y sonoros sorprendentemente dinámicos.

3. Ampliabilidad

Casi todos los ordenadores prestan atención a la necesidad de conectar periféricos tales como impresoras, joysticks y unidades de disco. Sin embargo, muchos de los más populares

ordenadores domésticos necesitan interfaces externas para que se les pueda conectar una simple impresora o un joystick.

El comprador no siempre prevé las necesidades futuras, pero un ordenador que incorpore un interfaz para impresora o un conector para joystick puede ser más barato a la larga que otro por el que tenga que pagar menos en el momento de la adquisición.

El CPC464 tiene incorporada una puerta de salida para impresora (tipo Centronics), un conector para dos joysticks, una salida de sonido estereofónico y una puerta de expansión a la que se puede conectar una unidad de disco (modelo DDI-1), un interfaz serie (modelo RS232C), un modem, un sintetizador de voz (modelo SSA2), un lápiz óptico, etc.

606 42 98

4. Sonidos

Las características sonoras de un ordenador determinan si 'suena a lata' o si puede producir una imitación aceptable de un instrumento musical electrónico.

El CPC464 está dotado de un generador de sonido de 3 canales y 8 octavas, que puede dar una calidad musical muy aceptable y permite el control completo de las envolventes de tono y de volumen. Además, el sonido se puede obtener en una configuración estereofónica, con canal izquierdo, canal derecho y canal común.

Esto permite programar el sonido para coordinarlo con el movimiento de objetos por la pantalla.

En último extremo, es usted quien tiene que decidir qué características le son más necesarias. Esperamos que las pruebe todas, para así aprovechar al máximo su ordenador.

¿Por qué no puede?

Sabiendo lo poderosa que es la tecnología actual, los usuarios se preguntan frecuentemente por qué una máquina tan avanzada como el CPC464 es incapaz de imitar las imágenes de un televisor de color. Por ejemplo, ¿por qué no puede un ordenador mostrar en la pantalla la imagen de una persona caminando con naturalidad? ¿Es que los ordenadores no son capaces de dar movimiento más que a unos burdos monigotes?

La respuesta es fácil, aunque compleja. No crea usted que la pantalla de su ordenador puede tener la misma riqueza de detalles que la de un televisor. Los televisores funcionan con una información 'analógica', que puede describir, con resolución casi ilimitada, todos los grados posibles de intensidad de todos los colores del espectro visible. En términos informáticos, esto representa que la 'memoria' necesaria para almacenar una imagen de televisión es unas veinte veces mayor que la que se le dedica a un ordenador doméstico.

Pero esto es sólo una parte del problema. Para dar movimiento a una imagen, hay que procesar toda esa información a gran velocidad (del orden de 50 veces por segundo). No es imposible, pero para ello se necesitan máquinas que cuestan unos cuantos miles de veces más que un ordenador doméstico, al menos en la actualidad.

Hasta que el precio de la memoria de alta capacidad baje drásticamente (lo que ocurrirá algún día), los ordenadores pequeños tienen que arreglárselas con una memoria relativamente pequeña para controlar la pantalla, lo que representa menor resolución y movimientos más espásticos. Optimizando el hardware y el softaware se puede paliar considerablemente estas limitaciones, pero de momento tendremos que seguir conformándonos con imágenes torpes que ni siquiera llegan a parecerse a las de una película de dibujos animados de regular calidad.

Ese teclado me suena ...

¿Por qué no puedo sentarme ante el ordenador y escribir una sencilla página de texto?

No se deje engañar por el hecho de que el ordenador se parece a una máquina de escribir electrónica con pantalla. La pantalla no es una hoja de papel electrónico: es una 'consola de órdenes', expresión técnica que quiere decir que es una vía de comunicación entre el usuario y el lenguaje de programación que está en la memoria de la máquina.

Mientras no se le mande otra cosa, el ordenador tratará de entender todo lo que se le comunique a través del teclado, pues supone que son instrucciones. Cuando usted pulsa [INTRO], el ordenador examina lo tecleado; si no es inteligible para el BASIC residente, lo rechaza y emite el mensaje:

Syntax error

Ahora bien, puede ocurrir que el programa actualmente presente en la memoria sea un procesador de textos; en tal caso usted podrá escribir las palabras que desee, pulsar [INTRO] y seguir escribiendo a sus anchas, exactamente igual que si el ordenador fuera una máquina de escribir que utilizara papel electrónico. Pero, para que esto sea posible, antes se debe haber cargado en la memoria un programa de proceso de textos.

El ordenador parece combinar aparatos que nos resultan familiares de tanto verlos en casa y en la oficina, como son la pantalla y el teclado. Pero recuerde que la similitud normalmente es superficial y que el ordenador es una combinación de elementos con personalidad propia.

¿Quién teme a la jerga?

Como ocurre con todas las ramas especializadas de la industria, la informática ha desarrollado una jerga propia para facilitar la comunicación de conceptos complejos que de otra forma requerirían muchas palabras de lenguaje ordinario. Esto no ocurre solamente en el campo de la alta tecnología; todos hemos topado con barreras de este tipo levantadas por los profesionales de distintas áreas.

En el caso de la informática, la confusión se debe más a la forma de utilizar las palabras que a las palabras en sí. (Y en castellano, a la imposibilidad de traducir ciertas palabras inglesas o a que no nos tomamos la molestia de intentarlo.) Casi todos los que llegan a familiarizarse con la terminología de la informática se esfuerzan por utilizar las palabras de la forma más directa posible, con objeto de minimizar la dificultad de la comunicación. Pero no crea que el lenguaje informático es lo mismo que el lenguaje literario; la informática no es literatura, sino una ciencia precisa; aparte la sintaxis, la estructura de la comunicación en informática es directa y sencilla, y de ningún modo confusa o ambigua. Como no es literatura, la informática no ha necesitado desarrollar una asignatura de 'comentario de textos' para analizar el significado exacto del significado de los programas.

Independientemente de si el significado de un programa es obvio o no lo es, hay muchos aspectos de la programación que se pueden calificar de embarullados o de elegantes; ahora que se está asentando el revuelo inicial causado por la aparición de los microordenadores, se está prestando mayor atención a los aspectos formales de la construcción de programas.

La informática está penetrando en las mentes de muchos jóvenes que valoran la precisión y la sencillez de las ideas y de la forma de comunicarlas. No hay demasiados niños que a los diez años sean ya abogados, pero sí muchos que a esa edad saben programar.

Lo básico de BASIC

Casi todos los ordenadores domésticos tienen incorporado el lenguaje BASIC, que hoy por hoy es lo más próximo al lenguaje ordinario. El nombre del lenguaje no implica ninguna limitación en cuanto a sus posibilidades; de hecho, se ha escrito programas extremadamente complejos y potentes en BASIC.

Sin embargo, es evidente que el nombre ha atraído a muchos pricipiantes, que esperaban encontrar en BASIC un buen punto de partida entre el laberinto de los lenguajes de programación; y esto ha contribuido en gran medida a su universalidad.

BASIC es un lenguaje informático que interpreta un reducido número de órdenes, como resultado de las cuales realiza operaciones con los datos según se va ejecutando el programa. A diferencia del vocabulario humano medio, que consta de entre 5000 y 8000 palabras (además de la conjugación de los verbos, plurales, diminutivos, etc.), BASIC tiene que arreglárselas con sólo un par de centenares. Los programas escritos en BASIC tienen que ajustarse a reglas muy estrictas en cuanto a la forma de utilizar esas palabras. La sintaxis es muy precisa; todo intento de comunicarse con el ordenador en lenguaje coloquial producirá el frío mensaje:

Syntax error

Esto no representa una limitación seria, ya que BASIC está diseñado para trabajar con números. Las palabras son básicamente una generalización de las operaciones matemáticas elementales (+, −, etc.). El concepto más importante que tienen que asimilar los principiantes es que el ordenador sólo puede manejar datos numéricos. Es decir, la información que se envía a la CPU (unidad central de proceso) tiene que estar en forma numérica.

Números

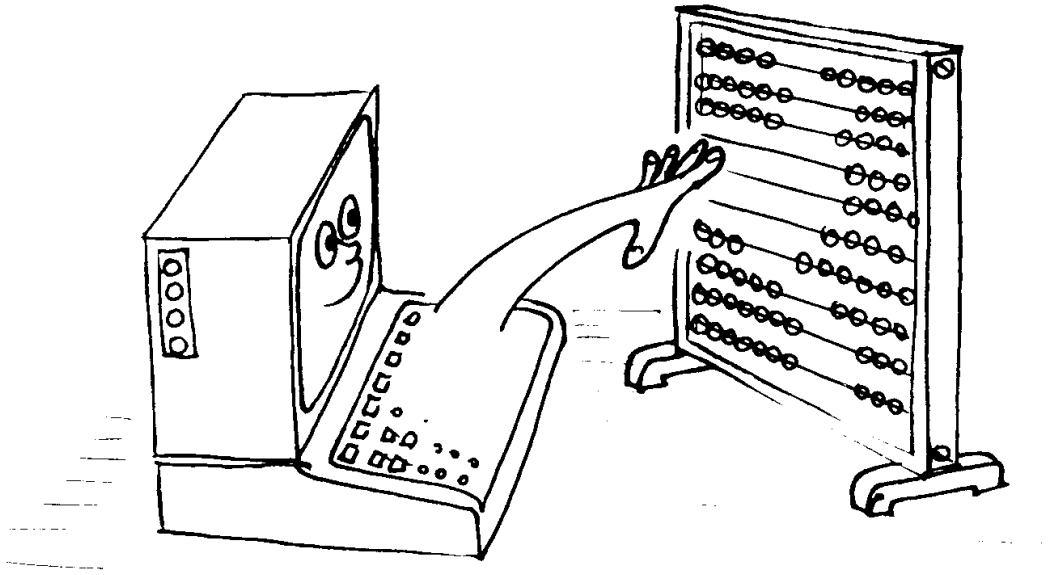
Si empleáramos un ordenador para almacenar las obras completas de Shakespeare, en todo el sistema no encontraríamos ni una sola letra o palabra. Todo elemento de información debe ser convertido a número antes de que el ordenador pueda reconocerlo y manipularlo.

BASIC interpreta las palabras como si fueran números; para procesarlas, el ordenador no dispone de más medios que la suma, la resta y la lógica de Boole, con la cual puede hacer comparaciones y seleccionar atributos; en otras palabras, averiguar si un número es menor o igual que otro, o realizar determinada tarea si un número cumple cierta condición.

A través del programa, el ordenador desmenuza todas las tareas en sus componentes más elementales.

Si todo esto le parece farragoso, está en lo cierto. Ha descubierto la primera y más importante verdad de la informática. Un ordenador es una herramienta que sirve para realizar tareas elementales y repetitivas a gran velocidad y con precisión absoluta. Así, BASIC interpreta las instrucciones que se le dan por medio del programa y las traduce al lenguaje que la CPU (unidad central de proceso) puede entender. La lógica del ordenador sólo distingue dos estados: sí y no; en notación binaria, 1 y 0; en notación de Boole, 'verdadero' y 'falso'. No hay estados intermedios del tipo 'quizá'.

El proceso de conmutación entre esos dos estados es la esencia del término 'digital'. En la naturaleza casi todos los procesos evolucionan gradualmente de un estado 'estable' a otro. En cambio, en la situación digital ideal, la transición de un estado a otro es instantánea. Sin embargo, las leyes de la física de semiconductores imponen un pequeño retardo, el retardo de propagación, que es el causante del tiempo que tarda el ordenador en procesar la información antes de dar la respuesta.



En todo caso, el ordenador tiene que esperar hasta que concluya una tarea antes de poder abordar la siguiente, lo que representa que siempre se le impone un retardo artificial. El proceso digital es 'blanco' o 'negro', y las fases de transición por los diversos grados de 'gris' no tienen ninguna importancia. En cambio, los procesos analógicos se realizan precisamente a través de esas fases de 'gris'.

Si la respuesta final sólo puede ser 0 o 1, no es posible que el resultado sea 'casi' correcto. El hecho de que los ordenadores cometan errores en ocasiones se debe a la limitación del tamaño de los números que pueden procesar; los números con demasiados dígitos tienen que ser 'comprimidos' para que encajen en el espacio disponible, lo que conduce a errores de redondeo.

En un mundo en el que sólo existen los dígitos 0 y 1, ¿cómo se cuenta más allá de 1?

Bits y bytes

Estamos habituados a entender los números solamente cuando están expresados en el sistema decimal, en el que el punto de referencia es el número 10 y en el que se dispone de 10 dígitos (del 0 al 9) para representar los números. El sistema binario es conceptualmente análogo al decimal, pero con sólo dos dígitos; las unidades con las que opera este sistema son los denominados *bits* (forma contraída de *BInary digiT*, dígito binario).

La relación entre bits y números decimales es sencilla y vamos a estudiarla.

Es habitual declarar el máximo número de bits que vamos a usar para representar los números, y añadir ceros por la izquierda para completar la expresión. Por ejemplo, el número decimal 7 se convierte en el binario

00111

en notación de 5 bits.

En el sistema binario, los dígitos se pueden considerar como indicadores dispuestos en columnas para especificar si una determinada potencia de 2 está presente o no lo está:

$$
\begin{aligned}
2^0 &= 1 \\
2^1 &= 2 = 2 = 2(2^0) \\
2^2 &= 4 = 2\times2 = 2(2^1) \\
2^3 &= 8 = 2\times2\times2 = 2(2^2) \\
2^4 &= 16 = 2\times2\times2\times2 = 2(2^3)
\end{aligned}
$$

de modo que las columnas tienen el siguiente aspecto:

2^4	2^3	2^2	2^1	2^0		
1	0	0	1	1		
(16 +	0 +	0 +	2 +	1)	=	19

Para abreviar la terminología al hablar de información binaria se introduce el concepto de 'byte', que es una sucesión de 8 bits. El mayor número que se puede representar con un

byte es 11111111 (binario) o 255 (decimal). Si contamos también el 00000000, resulta un total de 256 números distintos, que son datos perfectamente válidos para el ordenador.

Los ordenadores prefieren manejar los bits en grupos de 8. El número 256 es demasiado pequeño, y por eso, para poder controlar la memoria, se utilizan dos bytes que se interpretan en forma matricial. Cada posición de memoria tiene una dirección vertical y otra horizontal, y entre las dos caracterizan sin ambigüedad la posición. Por ejemplo, la siguiente matriz

	0	1	2	3	4	5	6	7	8	9
0										
1										
2										
3										
4										
5				1		1				
6										
7										
8										
9										

puede caracterizar hasta 10×10 elementos de información utilizando números de dirección comprendidos entre 0 y 9. El elemento almacenado en la posición 3,5 es un '1', lo mismo que el que está en la posición 5,5.

Análogamente, una matriz binaria de 256×256 puede caracterizar 65536 posiciones utilizando direcciones de 8 bits para los ejes vertical y horizontal de la matriz. Nuestros elementos iniciales, '0' y '1', han progresado hasta ser capaces de identificar 65536 elementos diferentes.

El siguiente nivel de abreviatura en terminología binaria es el K, que representa 1024 bytes. El número 1024 es la potencia de 2 más próxima a nuestro 'kilo' (1000), y explica por qué se dice que un ordenador es de 64K cuando en realidad tiene una memoria de 65536 bytes (64×1024).

Afortunadamente, el intérprete de BASIC se encarga de todas estas conversiones. Es perfectamente posible llegar a ser un programador aunque no se domine el sistema binario. No obstante, unos conocimientos del sistema binario pueden ayudar a desenmascarar la magia de ciertos números clave que inevitablemente nos encontramos en informática.

Vale la pena dedicar algún tiempo a entender el sistema binario y a comprender el por qué de números tales como el 255 o el 1024, pues parece evidente que estos números seguirán siendo el fundamento de la informática, al menos en el futuro previsible. La fiabilidad y sencillez que proporciona el sistema binario se seguirá imponiendo a la complejidad que implicaría cualquier otro sistema de numeración.

Sin embargo ...

Por sencilla y elegante que sea, la notación binaria induce a confusión a los humanos, porque no somos capaces de leer de un golpe de vista los números binarios. Hay algunos otros sistemas de numeración que los programadores utilizan con mayor facilidad. Un sistema muy utilizado en microordenadores es el hexadecimal (HEX). Su base es el número 16. La correspondencia entre dígitos decimales y hexadecimales es la siguiente:

Decimal

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Hexadecimal

0 1 2 3 4 5 6 7 8 9 A B C D E F

El sistema hexadecimal puede dividir los 8 bits de un byte en dos bloques de 4 bits, ya que 15 es un número de 4 bits: 1111 en binario. El primer bloque indica el número de unidades completas de '15'; el segundo bloque indica el 'resto'. Y aquí es donde empieza a aparecer la elegancia de los sistemas binario y hexadecimal. A continuación damos una tabla de conversión entre los tres sistemas para los primeros números:

Decimal	Binario	Hexadecimal
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10

Un número de 8 bits tal como el 11010110 (&D6 hex) puede ser subdividido en dos de 4 bits. A lo largo de todo este manual indicamos los números hexadecimales precediéndo los del signo &. El sistema hexadecimal es el más utilizado por los programadores que trabajan en ensamblador. El lenguaje ensamblador es la etapa más cercana a código de máquina a la que llegan normalmente los programadores; permite expresar las instrucciones en 'códigos nemotécnicos', que son algo más inteligibles que los códigos de máquina.

Para convertir un número hexadecimal en binario, lo primero que hay que hacer es obsevar el primer dígito, para ver cuántos grupos de '16' hay en el número, multiplicar por 16 y luego sumar al resultado el valor del segundo dígito. Así pues, no se deje llevar por la tentación de pensar que &D6 es 13+6, ni 136, porque en realidad es $(13\times16)+(6)=214$.

El proceso es el mismo que seguimos cuando leemos un número decimal: $89=(8\times10)+9$. Lo que ocurre es que nos resulta mucho más fácil multiplicar por 10 que por 16; es una cuestión de práctica.

Si usted ha conseguido llegar hasta aquí sin estar demasiado confuso, puede decir que está en el buen camino para llegar a entender los principios básicos del ordenador. Quizá se pregunte para qué nos metemos en todas estas complicaciones, y tiene razón. Un ordenador es un aparato que maneja ideas y conceptos muy sencillos; lo que tiene de especial es que realiza esas operaciones a una velocidad muy grande (millones de veces por segundo) y que tiene una enorme capacidad para recordar los datos, tanto los que se le han suministrado como los resultados intermedios de los muchos miles de sumas que realiza antes de poder dar la respuesta final.

Si desea seguir estudiando la teoría de los ordenadores, sepa que hay miles de libros dedicados a estos temas. Algunos no le aclararán en absoluto las ideas, sino todo lo contrario, pero otros en cambio le guiarán en el descubrimiento de la sencillez y las relaciones fundamentales que existen entre los sistemas de numeración y le ayudarán a comprender la forma en que el ordenador los maneja.

Parte 2: Más en concreto sobre el CPC464

En esta sección vamos a tratar algunas cuestiones específicas del CPC464. La información básica sobre estos temas se puede encontrar en el 'Curso de introducción' y en el capítulo titulado 'Lista completa de las palabras clave del BASIC de Amstrad'.

Temas tratados en esta sección:

* Juegos de caracteres
* ASCII
* Variables
* Lógica
* Caracteres definidos por el usuario
* Formatos de escritura
* Ventanas
* Interrupciones
* Datos
* Sonido

Caracteres

Cuando se ponga al teclado de su CPC464, no dé por supuesto que en la pantalla vayan a aparecer letras y números reconocibles. Ya hemos explicado que el ordenador no es una máquina de escribir. Lo que usted hace es pulsar una serie de conmutadores eléctricos; las señales eléctricas así producidas son traducidas por los circuitos internos para enviar a la pantalla grupos de puntos. Según sea la colocación de esos puntos, podremos reconocer letras, números u otros caracteres del 'juego de caracteres' del CPC464.

Algunos de estos caracteres no son accesibles directamente a través del teclado, sino mediante la instrucción **PRINT CHR$(**⟨número⟩**)**. Esto es así porque la información se almacena en el ordenador en unidades de 1 byte y, como vimos en la parte 1 de este capítulo, 1 byte puede tener 256 formas posibles. Dado que el ordenador tiene que dedicar al menos un byte a cada carácter almacenado, más vale que aprovechemos las 256 combinaciones posibles en lugar de conformarnos con los 96 'caracteres estándar' de las máquinas de escribir y desperdiciar los 160 restantes.

Los caracteres estándar son un subconjunto del juego total de caracteres. Su nombre en informática es 'caracteres ASCII', término derivado de 'American Standard Code for Infor-

mation Interchange' (código estándar norteamericano para el intercambio de la información). Se trata de un sistema diseñado para asegurar la compatibilidad de los datos transmitidos de un ordenador a otro. En el capítulo 7 hemos dado la lista de todos los caracteres ASCII, así como de los demás caracteres disponibles en el CPC464, junto a sus números de código.

Vamos a verlos

Seguramente ya no encierra ningún secreto para usted un programa tan sencillo como éste:

```
10 FOR n=32 TO 255
20 PRINT CHR$(n);
30 NEXT
```

con el que hacemos que el ordenador muestre en la pantalla el juego completo de caracteres. Pero estudiemos un poco la esencia de este programa.

Lo primero que debemos obsevar es que al ordenador no le hemos dicho **PRINT "abcdefghijklmn . . .",** sino **PRINT CHR$(n)**. En vez de mencionar explícitamente los caracteres hemos puesto en su lugar una 'variable'. Una *variable* es un elemento de información que varía según impongan las instrucciones del programa. (El nombre que hemos dado a la variable en este caso, **n**, es arbitrario; podríamos haber puesto cualquier letra o combinación de letras, cualquier cosa que no fuera una palabra reservada de BASIC.)

¿Cómo sabemos que es una variable?

Un número tal como el 5 es fijo, está entre el 4 y el 6 y no tiene nada de variable. El carácter **n** también es fijo: es una letra concreta del alfabeto.

Entonces, ¿cómo distingue el ordenador las variables de las constantes? Si hubiéramos querido decirle que considerara la **n** como letra del alfabeto, la habríamos escrito entre comillas: **"n"**. El ordenador habría respondido con el mensaje **Syntax error**, porque no entendería la línea **FOR "n"=32 TO 255**.

Al escribir la **n** sin comillas le hemos dicho al ordenador que **n** es una variable. La instrucción **FOR** de BASIC tiene que ir seguida del nombre de una variable, y el ordenador da por supuesto que lo que pongamos después de **FOR** es una variable.

También le hemos dicho al ordenador **n=32 TO 255**, con lo que hemos especificado el margen de variación de la variable: una sucesión que empieza en 32 y termina en 255.

Una vez declarada la variable, debemos decirle al ordenador qué tiene que hacer con ella; eso es lo que hace la línea 20:

```
20 PRINT CHR$(n);
```

Esta instrucción hace que el ordenador, cualquiera que sea el valor de **n**, busque en su memoria el carácter que corresponde a ese valor y lo escriba en la pantalla.

El punto y coma impide que el ordenador realice un retorno de carro y avance de línea después de escribir cada carácter. (De no ser así, los caracteres aparecerían uno debajo de otro en la primera columna de cada línea.)

La línea 30 le dice al ordenador que cuando haya concluido la tarea con el primer valor de **n** (que es el 32) debe retornar a la línea en la que está **FOR** y volver a hacer lo mismo con el siguiente (**NEXT**) valor de la variable **n**. Este mecanismo, denominado 'bucle', es fundamental para la programación. Gracias a él nos ahorramos el tener que escribir largas series de instrucciones similares. Usted aprenderá enseguida a incluirlo en sus programas.

Cuando el bucle **FOR ... NEXT** alcanza el límite superior del margen declarado (255), el bucle termina y el ordenador vuelve al modo directo y muestra el mensaje **Ready**. Este mensaje indica que el ordenador está preparado para recibir más instrucciones, una de las cuales podría ser **RUN**, para ejecutar otra vez el programa. El programa está almacenado en la memoria y lo seguirá estando hasta que le digamos al ordenador que lo borre o apaguemos la máquina.

Este programa ilustra un hecho esencial de la informática: todo lo que el ordenador hace está relacionado con números. El ordenador ha escrito el alfabeto y todos los demás caracteres usando como referencia para acceder a ellos un número. Cuando usted pulsa la tecla A, no le está pidiendo al ordenador que escriba esa letra, sino que busque en su memoria la información numérica que necesita para exhibir el carácter en la pantalla. La localización de esa información es función del código numérico que se activa al pulsar la tecla. (Cada carácter tiene un código asociado; la relación completa se da en el capítulo titulado 'Para su referencia ...'.)

Pero tampoco el hecho de visualizar el carácter tiene nada que ver con 'escribir', sino que es un proceso numérico.

Por ejemplo, el código ASCII de la letra A es el 65. El ordenador tampoco entiende el número 65, por lo que tiene que traducirlo de decimal a una forma que le resulte más familiar: el código de máquina. Ya hemos estudiado los principios de esta conversión en la primera parte de este capítulo.

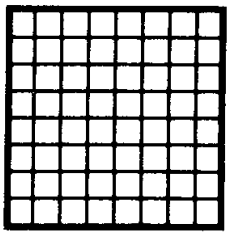
Al principio, la conversión de la notación decimal, a la que estamos tan acostumbrados, a notación hexadecimal nos parece muy difícil. Pensar en números decimales nos parece tan natural que cambiar de sistema de numeración es como cambiar de mano el tenedor y el cuchillo.

Para manejar la notación hexadecimal es necesario adquirir cierto grado de destreza mental. Una vez adquirida, se empieza a entender el por qué de muchos hechos informáticos y se hace patente la elegancia de la estructura de este sistema de numeración.

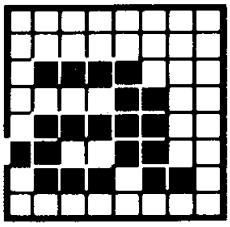
Si no conoce los sistemas binario y hexadecimal, le sugerimos que lea detenidamente la primera parte de este capítulo.

Cuando el ordenador ha convertido la pulsación de la tecla A en un número comprensible para él, busca en una zona de la memoria especificada por ese número, y en ella encuentra

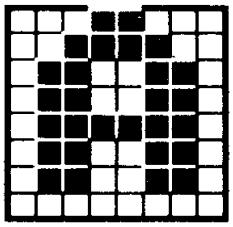
otra serie de números que describen la forma de la letra. El carácter que usted puede ver en la pantalla está formado por un bloque de datos, almacenado en la memoria en forma de 'matriz' numérica:



MATRIZ EN BLANCO
(RETICULA)



a MINÚSCULA



A MAYÚSCULA

Los elementos de la matriz son puntos dispuestos en filas columnas. El carácter se 'escribe' encendiendo y apagando correctamente los puntos. En el CPC464 cada matriz tiene 8 filas y 8 columnas. Si entre los 255 caracteres no encuentra el que necesita, usted mismo puede definirlo a su gusto con la instrucción **SYMBOL** que describiremos más adelante.

Los caracteres 'definidos por el usuario' se construyen especificando cada uno de los 64 puntos de la matriz. Las combinaciones posibles con 1.84467E+19, es decir, aproximadamente un 2 seguido de 19 ceros. Si a esto añadimos la posibilidad de agrupar caracteres para formar bloques más grandes, vemos que las posibilidades de generar gráficos sólo están limitadas por la imaginación del usuario.

Lógica

Una diferencia fundamental entre una calculadora y un ordenador radica en la capacidad de éste para manejar operaciones lógicas y tomar decisiones en instrucciones del tipo **IF ... THEN ...** (si ... entonces ...). Para ello, los operadores lógicos tratan los valores a los que son aplicados como grupos de bits y examinan y modifican los bits uno a uno. No es fácil describir las operaciones lógicas sin antes dar unas definiciones concisas.

Las dos mitades de una expresión lógica son los 'argumentos'. Una expresión lógica tiene la siguiente forma:

⟨argumento⟩ [⟨operador lógico⟩ ⟨argumento⟩]

donde ⟨argumento⟩ puede ser:

NOT ⟨argumento⟩
⟨expresión numérica⟩
⟨expresión de relación⟩
(⟨expresión lógica⟩)

Los dos argumentos de un operador lógico tienen que ser enteros; de lo contrario se produce el error número 6.

Los operadores lógicos (en orden de prioridad) y su efecto sobre los bits son los siguientes:

AND El resultado es 0 a menos que los bits de los dos argumentos sean 1.
OR El resultado es 1 a menos que los bits de los dos argumentos sean 0.
XOR El resultado es 1 a menos que los bits de los dos argumentos sean iguales.

AND es el operador lógico más utilizado.

```
PRINT 10 AND 10
```

da como resultado **10**.

```
PRINT 10 AND 12
```

da como resultado **8**.

```
PRINT 10 AND 1000
```

también da como resultado **8**.

Esto se debe a que los números **10** y **1000** han sido convertidos a su forma binaria:

```
      1010
1111101000
```

La operación **AND** comprueba los bits dos a dos. Si los dos bits de las dos filas son 1, el resultado es 1:

```
0000001000
```

y este número binario es el 8 decimal. El operador lógico **AND** se utiliza para detectar cuándo se cumplen simultáneamente dos condiciones. He aquí un ejemplo que explica cómo:

```
10 INPUT "Dia del mes: ",dia
20 INPUT "Mes número: ",mes
30 IF dia=25 AND mes=12 THEN 50
40 CLS:GOTO 10
50 PRINT "Feliz Navidad!"
```

OR funciona de forma análoga, pero da como resultado 1 a no ser que los bits de los dos operandos sean 0, en cuyo caso el resultado es 0. Aplicado a los mismos números que en el ejemplo de **AND**,

```
PRINT 1000 OR 10
 1002
```

Bit a bit:

```
      1010
1111101000
```

cuyo resultado es

```
1111101010
```

Un programa ejemplo:

```
10 CLS
20 INPUT "Mes número: ",mes
30 IF mes=12 OR mes=1 OR mes=2 THEN 50
40 GOTO 10
50 PRINT "Estamos en invierno!"
```

El operador **NOT** invierte todos los bits del argumento (convierte el 0 en 1, y viceversa):

```
10 CLS
20 INPUT "Mes número: ",mes
30 IF NOT(mes=6 OR mes=7 OR mes=8) THEN 50
40 GOTO 10
50 PRINT "No estamos en verano!"
```

Observe que se pueden combinar los operadores lógicos, sin más limitación que la longitud de la línea:

```
10 INPUT "Dia del mes: ",dia
20 INPUT "Mes número: ",mes
30 IF NOT(mes=12 OR mes=1) AND dia=29 THEN 50
40 CLS:GOTO 10
50 PRINT "No es ni diciembre ni enero, pero puede ser un año bisiesto"
```

El resultado de una expresión de relación es **−1** o **0**. La representación binaria del número **−1** es una sucesión de bits iguales a 1; para el **0**, todos los bits son 0. El resultado de una operación lógica entre argumentos de este tipo da **−1** (verdadero) o **0** (falso).

Añada las siguientes líneas al programa anterior:

```
60 PRINT NOT(mes=12 OR mes=1)
70 PRINT (mes=12 OR mes=1)
```

Si al ejecutar el programa introduce **29** para el día y **2** para el mes, obtendrá el mensaje de la línea 50, y las líneas 60 y 70 escribirán los resultados de las operaciones.

Finalmente, **XOR** (**OR** e**X**clusivo) produce como resultado 'verdadero' siempre que los argumentos sean diferentes.

A continuación resumimos todo lo dicho en una tabla, denominada *tabla de verdad*. Es una forma muy clara de ilustrar qué sucede en una operación lógica bit a bit.

Argumento A	1010
Argumento B	0110
Resultado de AND	0010
Resultado de OR	1110
Resultado de XOR	1100

Caracteres definidos por el usuario

Una de las primeras aplicaciones que el lector encontrará para los números binarios es el diseño de caracteres con la instrucción **SYMBOL**. Como sabemos, el carácter se diseña en una retícula de 8 por 8; cada una de las 8 filas se convierte en un número binario poniendo un 1 en lugar del pixel que debe ser iluminado, y un cero en lugar del pixel que debe quedar del color del papel. Los 8 números así obtenidos serán los parámetros de **SYMBOL**. Por ejemplo, para definir un carácter que representa una casa:

```
    *      = 00001000 = &08 =                8          = 8
  ****     = 00111100 = &3C =          32+16+8+4        = 60
 *    *    = 01000010 = &42 =     +64            +2     = 66
* *  * *   = 10100101 = &A5 = 128            +4   +1    = 165
*       *   = 10000001 = &81 = 128                 +1    = 129
* ** * *   = 10110101 = &B5 = 128    +32+16   +4   +1    = 181
* **   *   = 10110001 = &B1 = 128    +32+16        +1    = 177
******** = 11111111 = &FF = 128+64+32+16+8+4+2+1 = 255
```

la orden es

```
SYMBOL 240,8,60,66,165,129,181,177,255
```

o bien

```
SYMBOL 240,&08,&3C,&42,&A5,&81,&B5,&B1,&FF
```

o bien

```
SYMBOL 240,&X00001000,&X00111100,&X01000010,&X10100101,&X10000001,
          &X10110101,&X10110001,&X11111111
```

Para escribir el carácter así definido:

```
PRINT CHR$(240)
```

Finalmente, para agrupar caracteres se puede hacer

```
adosado$=CHR$(240)+CHR$(240)
PRINT adosado$
```

o bien

```
urbanizacion$=STRING$(15,240)
PRINT urbanizacion$
```

La imprenta ...

PRINT es la primera instrucción que se usa cuando se empieza a programar. En principio es muy sencilla, pero podemos complicarla considerablemente. En efecto, no basta con pedirle al ordenador que escriba, sino que también debemos decirle dónde y cómo.

Formato de escritura

La instrucción **PRINT** se puede utilizar de diversas formas. La más sencilla es poner a su derecha el elemento que se desea escribir, que puede ser un número, una cadena literal o el nombre de una variable:

```
PRINT 3
 3

PRINT "hola"
hola

a=5
PRINT a
 5

a$="prueba"
PRINT a$
prueba
```

En una instrucción **PRINT** se pueden poner varios elementos, intercalando entre ellos un separador, o bien **TAB** o **SPC**. Los separadores pueden ser la coma o el punto y coma. El punto y coma hace que el siguiente elemento se escriba inmediatamente a continuación del que se acaba de escribir; la coma provoca el salto a la siguiente zona de escritura. La anchura implícita de la zona de escritura es **13**, pero se la puede modificar con la orden **ZONE**:

```
PRINT 3;-4;5
 3 -4  5

PRINT "buenos","dias"
buenos          dias

PRINT 3,-4,5
 3              -4              5

ZONE 4
PRINT 3,-4,5
 3  -4   5
```

Observe que los números positivos llevan un espacio a la izquierda, mientras que en los negativos ese espacio está ocupado por el signo –. Todos los números llevan un espacio en blanco a la derecha. Las cadenas se escriben literalmente tal como están entre las comillas.

La función **SPC** lleva como parámetro una función numérica; 'escribe' tantos espacios como indique el valor de la expresión. Si el valor es negativo, se toma el 0; si es mayor que la anchura de la ventana actual se toma esa anchura:

```
PRINT SPC(5)"hola"
      hola

x=3
PRINT SPC(x*3)"hola"
            hola
```

TAB es similar, pero el número de espacios que escribe es el necesario para que el siguiente elemento se empiece a escribir en la columna especificada.

La ventana en la que se escribe es la número **0**, a no ser que se especifique otro número de canal (#) antes de la lista de elementos. Se pueden especificar otros canales para enviar la salida a otras ventanas. Los canales **8** y **9** están reservados para la impresora y la cinta, respectivamente. (Obsérvese que para 'escribir' en el canal #**9** se debe utilizar **WRITE** en lugar de **PRINT**.)

```
PRINT "hola"
hola                    – ventana 0

PRINT #0,"hola"
hola                    – también ventana 0

PRINT #4,"hola"
hola                    – ventana 4 (primera línea de la pantalla)

PRINT #8,"hola"
hola                    – en la impresora (si está conectada)
```

Con **TAB** y **SPC** se pueden controlar los formatos más sencillos, pero en cuanto el formato deseado sea un poco más complejo habrá que utilizar **PRINT USING** y una 'plantilla' adecuada. Una plantilla de formato es una expresión literal que contiene caracteres especiales, cada uno de los cuales especifica un formato determinado. Estos caracteres, denominados 'especificadores de formato', están explicados en detalle en la descripción de **PRINT USING** (capítulo 3). Aquí vamos a dar algunos ejemplos.

En primer lugar, he aquí los formatos disponibles para escribir cadenas literales:

" " escribe tantos caracteres como espacios haya en la plantilla, más dos:

```
PRINT USING "          ";"cadena de prueba"
cadena d
```

"!" escribe solamente el primer carácter de la cadena:

```
PRINT USING"!";"cadena de prueba"
c
```

Pero el formato literal más útil es seguramente "&". Con él se anula una función de BASIC por la cual, si una cadena es demasiado larga como para caber en la línea actual, el ordenador la escribe al principio de la línea siguiente. **PRINT USING "&"**; desactiva esa función. (En el siguiente ejemplo ponga **BORDER 0** para hacer visibles los bordes del papel.)

```
MODE 1:LOCATE 39,1:PRINT "demasiado"
                                                    – línea 1
demasiado
                                                    – línea 2
MODE 1:LOCATE 39,1:PRINT USING "&";"demasiado"
                                        de          – línea 1
masiado                                             – línea 2
```

Para los números se dispone de gran variedad de plantillas. La más sencilla es **PRINT USING** "#####", en la que cada # reserva espacio para un dígito:

```
PRINT USING "######";123
   123
```

La posición del punto decimal se puede especificar incluyendo un punto en la plantilla:

```
PRINT USING "####.#####";12.45
  12.45000
```

Los dígitos que quedan a la izquierda del punto decimal se pueden agrupar de tres en tres, separados por comas, si se pone una coma a la izquierda del punto en la plantilla:

```
PRINT USING "########,.####";123456.78
  123,456.7800
```

En el formato se pueden incluir los signos de dólar y de libra esterlina, para que aparezca siempre el signo monetario antes del primer dígito, aunque el número no llene el formato. Esto se consigue poniendo **$$** en la plantilla:

```
PRINT USING "$$##";7
  $7

PRINT USING "$$##";351
$351

PRINT USING "$$####,.##";1234.567
 $1,234.57
```

Observe cómo se ha redondeado este último número.

Se puede rellenar el espacio sobrante por la izquierda con asteriscos poniendo ** en la plantilla:

```
PRINT USING "**####.#";12.22
****12.2
```

Esto último se puede combinar con los signos monetarios poniendo **$ en la plantilla.

Un signo + al principio de la plantilla especifica que siempre se debe escribir el signo del número a su izquierda. Si el + está al final de la plantilla, el signo se describe a la derecha del número.

El signo – sólo se puede poner al final de la plantilla; especifica que se debe poner el signo – a la derecha del número si éste es negativo.

```
PRINT USING "+##";12
+12

PRINT USING "+##";-12
-12

PRINT USING "##+";12
12+

PRINT USING "##-";-12
12-

PRINT USING "##-";12
12
```

La inclusión de "↑↑↑↑" en la plantilla hace que el número se escriba en forma exponencial:

```
PRINT USING "###.##↑↑↑↑";123.45
 12.35E+01
```

En cualquier caso, si el número es demasiado grande como para caber en el formato especificado, el número no se trunca, sino que se lo escribe entero, precedido de un signo % para indicar lo que ha ocurrido:

```
PRINT USING "####";123456
%123456
```

Ventanas

El BASIC del CPC464 permite la definición y el control de hasta ocho ventanas de texto. Todas las acciones de control de textos se pueden referir a cualquiera de estas ventanas.

Las ventanas se definen con la orden **WINDOW**, que va seguida de 5 parámetros. El primero es opcional y especifica el número de la ventana que se va a definir; si se lo omite, el ordenador supone el **0**, que es el canal por el que el BASIC emite sus mensajes normales (**Ready**, errores, etc.). Antes del número se pone el signo # para indicar que se está dando un número de canal. Los otros cuatro parámetros especifican los extremos izquierdo, derecho, superior e inferior de la ventana. Como son números de fila y de columna, pueden estar comprendidos entre 1 y 80 los dos primeros, y entre 1 y 25 los dos últimos.

El siguiente ejemplo define la ventana (canal) número 4, la cual se extiende desde la columna 7 hasta la 31, y desde la fila 6 hasta la 18. Reinicialice la máquina y luego escriba:

```
WINDOW #4,7,31,6,18
```

No parece que haya ocurrido nada, así que escriba lo siguiente:

```
INK 3,9
PAPER #4,3
CLS #4
```

En la pantalla ha aparecido un gran rectángulo verde, que es la ventana número 4. Este último ejemplo demuestra que las instrucciones **PAPER** y **CLS** se pueden referir a cualquiera de las ocho ventanas especificando el número de canal; su omisión hace que la orden actúe sobre la ventana número **0**.

Las órdenes para las que se puede especificar número de ventana son las siguientes:

CLS, INPUT, LINE INPUT, LIST, LOCATE, PAPER, PEN, POS, PRINT, TAG, TAGOFF, VPOS, WINDOW, WRITE.

La ventana verde que hemos creado habrá borrado parte del texto que teníamos en la pantalla, que había sido enviado a la ventana número **0**.

El texto se puede escribir en cualquier ventana especificando el número de canal en la instrucción **PRINT**:

```
PRINT #4,"estoy en la ventana 4"
```

Estas palabras han aparecido en el extremo superior de la ventana verde, no en la línea siguiente de la pantalla, que es lo que habría ocurrido con:

```
PRINT "estoy en la ventana 0"
```

Observe que al escribir esta última orden el texto ha invadido parte de la ventana verde.

Si usted desea confinar todos los mensajes de BASIC a la ventana número **4**, intercámbiela con la implícita (la número **0**) mediante la orden:

```
WINDOW SWAP 0,4
```

El mensaje **Ready** ha aparecido ahora en la ventana verde y el cursor está debajo de él. Escriba lo siguiente:

```
PRINT #4,"estoy en la ventana 4"
```

El texto ha aparecido debajo de la orden **WINDOW SWAP**, en lo que antes era la ventana **0** y ahora es la **4**. Estos ejemplos demuestran que el ordenador recuerda la posición de escritura de cada ventana incluso después de un intercambio. Escriba lo siguiente:

```
LOCATE #4,20,1
PRINT "estoy en la ventana 0"
PRINT #4,"estoy en la ventana 4"
```

Mientras no se ejecuta una orden **WINDOW**, todas las ocho ventanas coinciden con la pantalla entera. Lo mismo ocurre después de una orden **MODE**. Así pues, si después de jugar con las ventanas se encuentra con que el cursor está en una muy pequeña, puede salir del lío escribiendo **MODE 1**:

```
MODE 1
WINDOW 20,21,7,18
MO
DE
 1
```

El ordenador se ha visto obligado a partir la palabra **MODE**, pero la instrucción funciona igual. (No olvide dejar un espacio entre la **E** y el **1**.)

Ahora que ya tiene una idea de cómo funcionan las ventanas, escriba y pruebe el siguiente programa:

```
10 MODE 0
20 FOR n=0 TO 7
30 WINDOW #n,n+1,n+6,n+1,n+6
40 PAPER #n,n+4
50 CLS #n
60 FOR c=1 TO 200:NEXT c
70 NEXT n
```

Este programa crea 8 ventanas que se solapan y las borra con un color de papel diferente para cada una.

Cuando termine el programa y haya aparecido **Ready**, pulse [INTRO] varias veces para observar cómo afecta el desplazamiento de la ventana **0** a los bloques de colores. Sin embargo, aunque el contenido de las ventanas se desplace, las ventanas en sí siguen estando en el mismo sitio. Compruébelo escribiendo:

```
CLS #4
```

Fíjese también en los diferentes efectos de las siguienes órdenes:

```
LIST
LIST #4
LIST #3
```

Otra característica interesante de la orden **WINDOW** es que también admite los números que especifican los bordes izquierdo y derecho en orden inverso; es decir, aunque el primer parámetro sea mayor que el segundo, BASIC los interpreta en el orden correcto. Análogamente ocurre con los bordes inferior y superior.

```
10 MODE 0
20 a=1+RND*19:b=1+RND*19
30 c=1+RND*24:d=1+RND*24
40 e=RND*15
50 WINDOW a,b,c,d
60 PAPER e:CLS
70 GOTO 20
```

Interrupciones

Por si todavía no lo ha leído en otro lugar de este manual, le diremos que una de las principales innovaciones del software de los ordenadores Amstrad es su capacidad de gestión de las interrupciones desde BASIC; esto hace que el BASIC de Amstrad sea capaz de realizar

acciones simultáneas controladas por programa. Esto es lo que se suele denominar 'multitarea', y se programa con las instrucciones **AFTER** y **EVERY**.

Esta misma habilidad queda demostrada por la forma en que se controla el sonido a través de recursos tales como las colas de sonido y la sincronización de canales.

De todo lo que tenga que ver con la medida del tiempo se encarga el cronómetro patrón del sistema, que es un circuito controlado por un cristal de cuarzo. Este reloj se encarga de cronometrar y de sincronizar los procesos que tienen lugar dentro del ordenador: desde el barrido de la pantalla hasta el envío de impulsos al microprocesador. Toda función del hardware que tenga que ver con el tiempo depende del reloj patrón.

Las instrucciones **AFTER** (después de) y **EVERY** (cada) hacen precisamente lo que su nombre sugiere. Así, **AFTER** invoca una determinada subrutina cuando ha transcurrido el tiempo especificado.

El CPC464 mantiene un reloj de tiempo real. La orden **AFTER** permite que un programa de BASIC realice una tarea en un instante futuro especificado. Existen cuatro temporizadores de retardo, cada uno de los cuales puede tener una subrutina asociada.

Cuando el tiempo especificado ha transcurrido, la rutina se ejecuta automáticamente, exactamente igual que si el ordenador hubiera encontrado un **GOSUB** en la línea actual del programa. Cuando la rutina se acaba, con el **RETURN** habitual, el programa continúa a partir del punto en que fue interrumpido.

La orden **EVERY** permite que el programa de BASIC ejecute subrutinas a intervalos de tiempo regulares. También en este caso se dispone de cuatro temporizadores, a cada uno de los cuales se puede asignar una subrutina distinta.

Los temporizadores tienen prioridades de interrupción diferentes, lo que tiene importancia cuando varios de ellos compiten por interrumpir el programa en un momento dado. El temporizador 3 tiene la máxima prioridad, y el 0 la mínima (véase el capítulo 'Para su referencia . . .').

```
10 MODE 1:n=14:x=RND*400
20 AFTER x,3 GOSUB 90
30 EVERY 25,2 GOSUB 150
40 EVERY 10,1 GOSUB 160
50 PRINT"Pruebe sus reflejos:"
60 PRINT"pulse la barra espaciadora"
70 PRINT"cuando yo le diga."
80 IF ind=1 THEN END ELSE 80
90 x=REMAIN(2)
100 SOUND 129,20:PRINT"AHORA":t=TIME
110 IF INKEY(47)=-1 THEN 110
120 PRINT"Ha tardado";
130 PRINT(TIME-t)/300;"segundos"
140 ind=1:RETURN
150 SOUND 1,0,50:PRINT".";:RETURN
160 n=n+1:IF n>26 THEN n=14
170 INK 1,n:RETURN
```

Las órdenes **AFTER** y **EVERY** se pueden ejecutar en cualquier lugar del programa. Su efecto es reelegir la rutina asociada y poner a cero el temporizador correspondiente. Los temporizadores están compartidos por **AFTER** y **EVERY**, de modo que una orden **AFTER** cancela la anterior **EVERY** referida al mismo temporizador, y viceversa.

Las órdenes **DI** y **EI** inhiben y habilitan, respectivamente, las interrupciones. Esto sirve para permitir que cierta interrupción se procese sin ser interrumpida por otra más prioritaria. La orden **REMAIN** da el tiempo que queda en un temporizador y lo desactiva.

Listas de datos

Si un programa necesita que se le suministre siempre la misma información cada vez que se lo ejecuta, es preferible hacer que esa información esté contenida en el propio programa para no tener que teclearla con cada ejecución. Esto es posible gracias al par de instrucciones **READ/DATA**. La orden **READ** es similar a **INPUT** en el sentido de que asigna valores a variables, pero en lugar de captar los valores por el teclado los lee en listas encabezadas por la palabra **DATA**. Los dos ejemplos siguientes ilustran la diferencia:

```
10 INPUT "escriba 3 numeros separados por comas: "a,b,c
20 PRINT"los numeros son";a;"y";b;"y";c
run

10 READ a,b,c
20 PRINT"los numeros son";a;"y";b;"y";c
30 DATA 12,14,21
run
```

Los datos contenidos en las listas **DATA** van separados por comas, lo mismo que los que se teclean en respuesta a una instrucción **INPUT**.

Las listas **DATA** pueden contener, no sólo números, sino también cadenas literales constantes:

```
10 DIM a$(11)
20 FOR i=0 TO 11
30 READ a$(i)
40 NEXT
50 FOR i=0 TO 11
60 PRINT a$(i);" ";
70 NEXT
80 DATA Cuentan,de,un,sabio,que,un,dia,tan,pobre,y,misero,estaba
run
```

Observe que no es necesario encerrar entre comillas las cadenas que se ponen en las listas **DATA**. Las comillas en este caso son opcionales (también lo son en **INPUT**). El único caso en que son necesarias es cuando una cadena contiene una coma; si no se la pusiera entre comillas, la instrucción **READ** se detendría en la coma y no leería el resto de la cadena.

```
10 READ a$
20 WHILE a$<>"*"
30 PRINT a$
40 READ a$
50 WEND
60 DATA los dias de la semana son lunes, martes, miércoles, jueves,
   viernes y sábado
70 DATA "los dias de la semana son lunes, martes, miércoles, jueves
   viernes y sábado"
80 DATA *
run
```

La cadena de la línea 60 contiene comas, y por lo tanto cada tramo será leído por **READ** y escrito por separado. En cambio, la de la línea 70 ha sido delimitada por comillas y **READ** la lee de una sola vez.

El ejemplo anterior demuestra que los datos pueden estar distribuidos en varias líneas (60, 70, 80, . . .). Otro detalle no tan evidente es que las líneas **DATA** pueden estar en cualquier lugar del programa, antes o después de las instrucciones **READ** que leen la información.

Si un programa contiene varias instrucciones **READ**, cada una continúa leyendo donde terminó la anterior:

```
10 DATA 123,456,789,321,654,2343
20 FOR i=1 TO 5
30 READ num
40 total=total+num
50 NEXT
60 READ total2
70 IF total=total2 THEN PRINT "los datos son correctos" ELSE PRINT
   "hay un error en los datos"
run
```

Modifique uno de los primeros cinco números de la lista de la línea 10 y ejecute otra vez el programa. Esta técnica de poner al final de la lista un dato adicional que sea la suma de los anteriores constituye un buen método de detección de errores en los datos, especialmente si los datos son muchos. Se le llama 'suma de verificación' (*checksum*).

Si un programa requiere datos mixtos (cadenas y números), se los puede combinar en una misma lista **DATA**, a condición de que **READ** los lea en el orden correcto. Por ejemplo, si una lista **DATA** contiene dos números seguidos de una cadena, la instrucción **READ** tendrá que leer primero dos variables numéricas y luego una literal:

```
10 DIM a(5),b(5),s$(5)
20 FOR i=1 TO 5
30 READ a(i),b(i),s$(i)
40 NEXT
50 DATA 1,7,Alfredo,3,9,Juan,2,2,Enrique,4,6,Pedro,9,1,Manuel
60 FOR i=1 TO 5
70 PRINT s$(i),":";a(i)*b(i)
80 NEXT
```

También se puede separar los datos en dos listas distintas:

```
10 DIM a(5),b(5),s$(5)
20 FOR i=1 TO 5
30 READ a(i),b(i)
40 NEXT
50 FOR i=1 TO 5
60 READ s$(i)
70 NEXT
80 DATA 1,7,3,9,2,2,4,6,9,1
90 DATA Alfredo,Juan,Enrique,Pedro,Manuel
100 FOR i=1 TO 5
110 PRINT s$(i),":";a(i)*b(i)
120 NEXT
```

Si ahora cambiamos el límite superior de la línea 20:

```
20 FOR i=1 TO 4
```

los dos primeros intentos de leer cadenas darán '**9**' y '**1**'. Estos valores son cadenas válidas, pero el programa no funciona como esperábamos. Una forma de asegurar que el programa funcione correctamente consiste en incluir las líneas:

```
15 RESTORE 80
45 RESTORE 90
```

La orden **RESTORE** hace que el puntero de datos 'apunte' hacia el principio de la línea especificada; por lo tanto se la puede incluir en una instrucción condicional para que el bloque de datos leídos dependa del resultado de alguna comprobación. Por ejemplo, en un juego que esté previsto para varios niveles de destreza, los datos de cada nivel pueden ser seleccionados mediante una variable adecuada. Veamos un ejemplo:

```
1000 REM sección para dibujar la pantalla
1010 IF nivel=1 THEN RESTORE 2010
1020 IF nivel=2 THEN RESTORE 2510
1030 IF nivel=3 THEN RESTORE 3010
1040 FOR y=1 TO 25
```

```
1050 FOR x=1 TO 40
1060 READ car
1070 LOCATE x,y:PRINT CHR$(car);
1080 NEXT x,y
...
2000 REM DATA para pantalla 1
2010 DATA 200,190,244,244,210,   ...   etc.
...
2500 REM DATA para pantalla 2
2510 DATA 100,103,245,243,251,   ...   etc.
...
3000 REM DATA para pantalla 3
3010 DATA 190,191,192,193,194,   ...   etc.
```

Otra aplicación típica de las instrucciones **DATA/READ/RESTORE** es un programa que interprete una melodía. Los periodos de tono se pueden poner en listas data; **RESTORE** puede hacer que se repita una sección moviendo el puntero de datos hacia atrás:

```
10 FOR i=1 TO 3
20 RESTORE 100
30 READ nota
40 WHILE nota <>-1
50 SOUND 1,nota,35
60 READ nota
70 WEND
80 NEXT
90 SOUND 1,142,100
100 DATA 95,95,142,127,119,106
110 DATA 95,95,119,95,95,119,95
120 DATA 95,142,119,142,179,119
130 DATA 142,142,106,119,127,-1
run
```

El sonido de la música

De todas las instrucciones del CPC464, es posible que las de sonido y envolventes le parezcan las más imponentes a primera vista. Y sin embargo, con un poco de práctica, muy pronto podrá programar ruidos de diversos tipos e incluso melodías con armonía.

Empecemos por analizar las cuatro primeras partes de la instrucción **SOUND**, que son las siguientes: situación de canales, periodo de tono, duración de la nota y volumen. Lo primero que hay que saber es en qué margen de valores puede estar cada uno de estos parámetros.

Aplazaremos el primero de momento porque es el más complicado. El segundo, periodo de tono, puede tener cualquier valor entero comprendido entre 0 y 4095. Sin embargo sólo algunos de estos valores producen notas reconocibles: son los que están relacionados en la parte 5 del capítulo 'Para su referencia ...'. Por ejemplo, el número 478 produce la nota DO media; el 506 produce la nota SI inmediatamente anterior; los valores 479 a 505 producen cada uno un tono diferente, pero ninguno de ellos corresponde a la escala del piano. Si el parámetro es 0, no se produce ningún tono; esto es útil cuando se está generando ruido.

El tercer parámetro de la orden **SOUND** especifica la duración de la nota en unidades de centésimas de segundo. Su valor puede estar normalmente entre 1 y 32767. Si es 0, la duración queda controlada por la 'envolvente' (de la que hablaremos más adelante). Si el parámetro es negativo, su valor obsoluto indica cuántas veces se va a repetir la envolvente; por ejemplo, −3 significa 'repetir la envolvente tres veces'.

El cuarto parámetro especifica el volumen. Puede valer entre 0 y 15; el valor implícito es 12, que es el que el ordenador supone si no se especifica otra cosa. En los sonidos sencillos que hemos conocido hasta ahora, el volumen ha permanecido constante durante el tiempo en que ha sonado cada nota. Sin embargo, cuando se utiliza una 'envolvente de volumen' para hacer que éste deje de ser constante, el parámetro 'volumen' de **SOUND** se considera como volumen inicial de la nota.

Bueno, vamos con el parámetro de situación de canales. Quizá sepa ya el lector que el significado de este parámetro depende del valor de sus bits; para entenderlo necesitará saber algo acerca de los números binarios (parte 1 de este capítulo).

El sonido se puede generar en tres canales distintos. Si el ordenador está conectado a un amplificador estereofónico, un canal será el derecho, otro el izquierdo y otro común a ambos o central. Para especificar en qué canal o canales debe sonar una nota se utilizan los siguientes números:

1 canal A
2 canal B
4 canal C

Para envíar el sonido a varios canales, se suman los números correspondientes. Por ejemplo, para que la nota suene en los canales A y C, el parámetro debe ser 1+4=5.

```
SOUND 5,284
```

Se preguntará el lector por qué el número del canal C es 4, y no 3. Observe que estos tres números son potencias de 2(1=2↑0, 2=2↑1, 4=2↑2) y se combinan para dar un número binario. Si imaginamos un número binario de tres dígitos, cada uno de ellos se puede utilizar para indicar si el canal correspondiente debe estar conectado o desconectado. En el ejemplo anterior, 5 en decimal es equivalente a **1*4+0*2+1*1**, es decir, 101 en binario. Si a los dígitos de este número binario les ponemos las etiquetas C, B y A, tenemos

C	B	A
1	0	1

de forma que C y A están conectados, mientras que B está desconectado. Si quisiéramos que la nota sonase en los canales A y B, el número tendría que ser

C	B	A
0	1	1

En decimal: **0*4+1*2+1*1=3**. La orden **SOUND** sería

```
SOUND 3,284
```

Naturalmente, este número tiene que coincidir con el que se obtendría sumando los valores correspondientes a los dos canales: 1+2=3 (recuerde que A=1, B=2, C=4).

Si después de todo no ha entendido cómo funciona esto en binario, no se preocupe. Le basta con saber que las combinaciones de canales se programan sumando los números de selección de los canales deseados.

Lamentablemente (o afortunadamente, según cómo se mire), todavía podemos sumar otros números a este parámetro. Así, los números 8, 16 y 32 especifican que el sonido debe sincronizarse con otro canal (A, B o C, respectivamente). Ahora hace falta saber qué es eso de 'sincronizar con un canal'. Pues bien, los sonidos que hemos generado hasta ahora han ido directamente al canal especificado. Escriba lo siguiente:

```
SOUND 1,284,2000
SOUND 1,90,200
```

A menos que sea usted un mecanógrafo muy lento, habrá tenido tiempo de escribir la segunda orden antes de que se extinguiera el primer sonido. Esto ocurre porque el sistema de sonido puede guardar hasta cinco órdenes en cada una de las colas de los tres canales. Si queremos que suene una nota por el canal A y luego dos notas simultáneamente en los canales A y B, necesitamos un forma de indicar al ordenador que no debe ejecutar la nota del canal B mientras no haya terminado la primera del A. En esto consiste la sincronización de canales. Hay dos formas de conseguirla:

```
SOUND 1,200,1000
SOUND 3,90,200
```

En este ejemplo hemos dirigido la segunda nota a A y B, y por lo tanto no puede sonar mientras no haya concluido la primera. La limitación de este método es que el sonido que se envía a varios canales tiene que ser igual en todos ellos (en este caso, **,90,200** era igual para el A que para el B). El otro método es el siguiente:

```
SOUND 1,200,2000
SOUND 1+16,90,200
SOUND 2+8,140,400
```

Aquí hemos hecho que la segunda nota de A se sincronice con el sonido de B (y que éste se sincronice con el canal A). La ventaja de este método es evidente: las notas sincronizadas

pueden ser (y en este caso son) diferentes. Estos números de sicronización también son interpretables bit a bit:

8=2↑3, 16=2↑4, 32=2↑5

Así, el número de situación de canales se puede considerar como número binario cuyos dígitos tienen el siguiente significado:

Sincr. con C Sumar 32	Sincr. con B Sumar 16	Sincr. con A Sumar 8	Sonar en C Sumar 4	Sonar en B Sumar 2	Sonar en A Sumar 1
0	0	1	1	0	0

Éste es el número binario 1100, equivalente a 8+4 en decimal.

Así pues, el número de situación de canales 12 ordena a la máquina que haga sonar una nota en el canal C y que espere por una nota que debe sonar en el canal A y que ha sido marcada para sincronizarla con el C.

Si ahora sumamos 64 (=2↑6) al parámetro, estamos indicando que la nota debe ser retenida. Esto significa que no va a sonar mientras no la liberemos con la orden **RELEASE**.

Finalmente, si sumamos 128 (=2↑7), borramos la cola de sonido del canal especificado.

Por ejemplo, si hemos ordenado un sonido que va a durar un buen rato, podemos anularlo borrando la cola del canal correspondiente:

```
SOUND 1,248,30000        (esta nota dura 5 minutos)
SOUND 1+128,0            (pero esta la detiene)
```

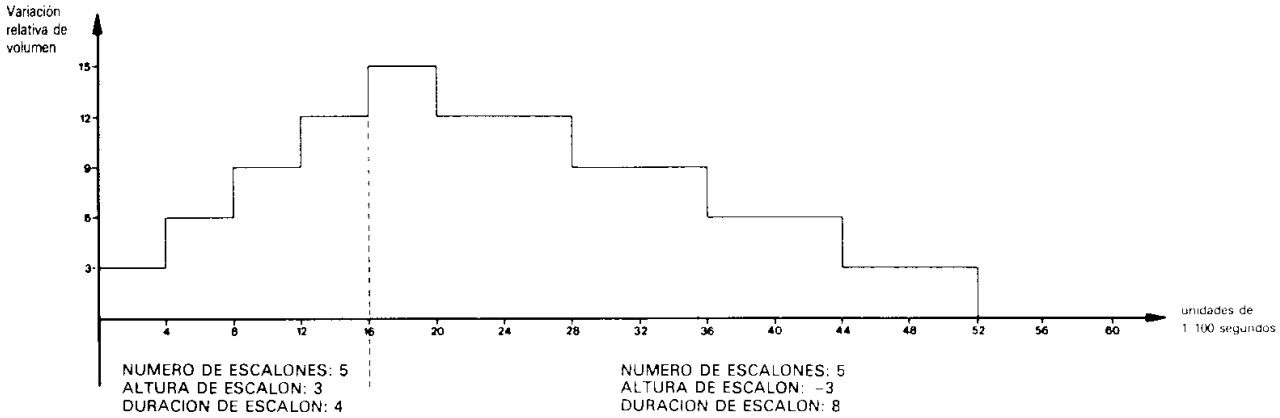
Cuando se esté en modo directo, la forma más rápida de interrumpir un sonido es pulsar [←BORR] al principio de la línea; el pitido que así se produce borra todas las colas de sonido.

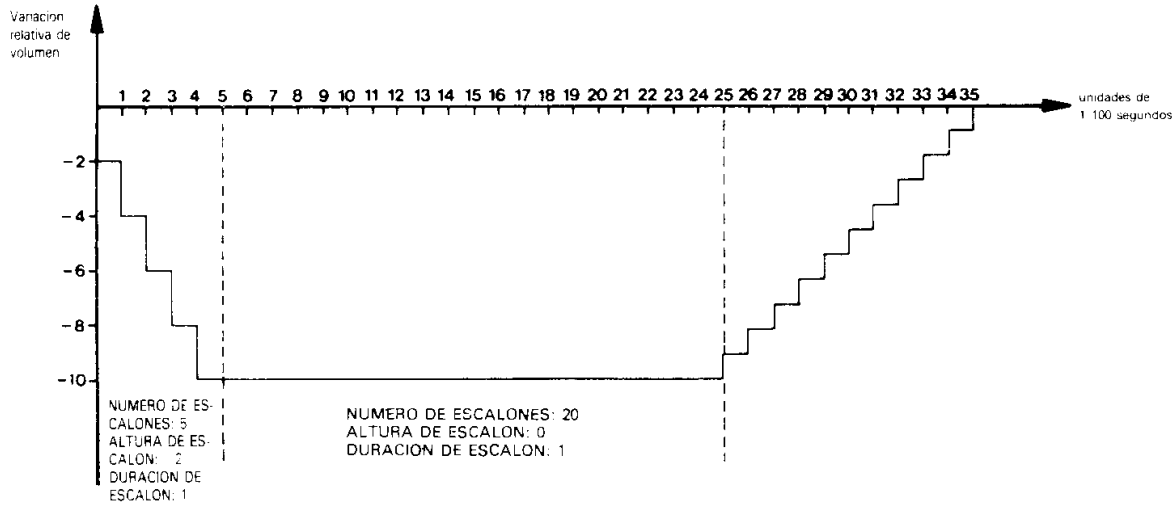
Ahora que ya sabemos enviar sonido a cualquier combinación de canales (con sincronización, si es necesario), vamos a intentar producir algo más agradable que los molestos pitidos que produce una orden **SOUND** en esta forma sencilla. Lo haremos dotando al sonido de una envolvente: una gráfica que define cómo evoluciona la intensidad del sonido a lo largo del tiempo. Las notas producidas por los instrumentos musicales tienen una fase inicial de ataque en la que el volumen sube muy deprisa; después se mantiene a un nivel algo inferior hasta que finalmente decae gradualmente a cero. En el ordenador se puede dar una envolvente de este tipo a las notas producidas por la orden **SOUND**. La instrucción con la que se programa la envolvente es **ENV**. Veamos un ejemplo sencillo:

```
ENV 1,5,3,4,5,-3,8
SOUND 1,284,0,0,1
```

La instrucción **ENV** se debe ejecutar antes que la **SOUND** que la utiliza. Para invocar una envolvente en una instrucción **SOUND** se pone como quinto parámetro de éste el número

de referencia de la envolvente, en este caso el 1, que es el número con que se la creó en la instrucción **ENV**. Los parámetros de **ENV** describen tanto la duración como el volumen de la nota, de modo que en **SOUND** podemos poner **0** en lugar de los datos de duración y volumen. La envolvente del ejemplo anterior hace que la nota crezca en 5 etapas, en cada una de las cuales el volumen aumenta en 3 unidades y cuya duración es de 4 centésimas de segundo. Después se especifica que el volumen debe decaer en 5 etapas, −3 unidades en cada una, siendo la duración de cada etapa 8 centésimas de segundo. Es decir, el primer parámetro de **ENV** es el número de referencia de la envolvente y va seguido por grupos de tres números. Dentro de cada grupo, el primer parámetro es el número de escalones de variación del volumen; el segundo, la amplitud de esos escalones; y el tercero, su duración. La duración total de cada sección será igual al producto del primer parámetro (número de escalones) por el tercero (duración de cada escalón). El aumento o disminución total del volumen es igual al producto del primer parámetro por el segundo (variación del volumen por escalón). La duración total de una envolvente es la suma de las correspondientes a las secciones que la integran.





En el ejemplo anterior, **SOUND** fijaba en **0** el volumen inicial de la nota. Pero esto no tiene que ser necesariamente así. En el siguiente ejemplo el volumen decrece desde el valor inicial **15** y luego vuelve a subir:

```
ENV 2,5,-2,1,20,0,1,10,1,1
SOUND 1,284,0,15,2
```

El número de esta envolvente es el **2**. Consta de tres secciones. En la primera el volumen se reduce en **5** escalones de −2; es decir, varía a través de 5 etapas y en cada una se reduce en 2 unidades. La duración de cada etapa es de **1** centésima de segundo. La segunda sección tiene **20** etapas en las que no varía el volumen (**0**) y cada una de las cuales dura **1** centésima de segundo. Finalmente, la tercera sección consta de 10 etapas, con un incremento de volumen de **1** unidad y duración de **1** centésima de segundo cada una.

La instrucción **SOUND** especifica un volumen inicial de **15**; al final de la primera sección ha decrecido hasta 5; se mantiene a ese nivel durante 20 centésimas de segundo y luego vuelve a crecer hasta alcanzar el nivel 15.

Para mejor visualizar y diseñar la forma de las envolventes conviene dibujarlas en un papel milimetrado y leer en él los valores de los parámetros requeridos. En las dos figuras de la página anterior mostramos las dos envolventes definidas hasta ahora.

El máximo número de secciones que se puede incluir en una envolvente es 5; como cada sección requiere 3 parámetros, la instrucción **ENV** puede llegar a tener 16 parámetros en total, contando el primero, que indica cuál de las 15 envolventes posibles se está definiendo. Podemos imaginar los números de volumen como dispuestos en círculo; si vamos subiendo de nivel y sobrepasamos el 15, volvemos al 0; análogamente, si intentamos descender por debajo del 0, volvemos al 15. Por ejemplo, en

```
ENV 3,9,5,20
SOUND 1,284,0,0,3
```

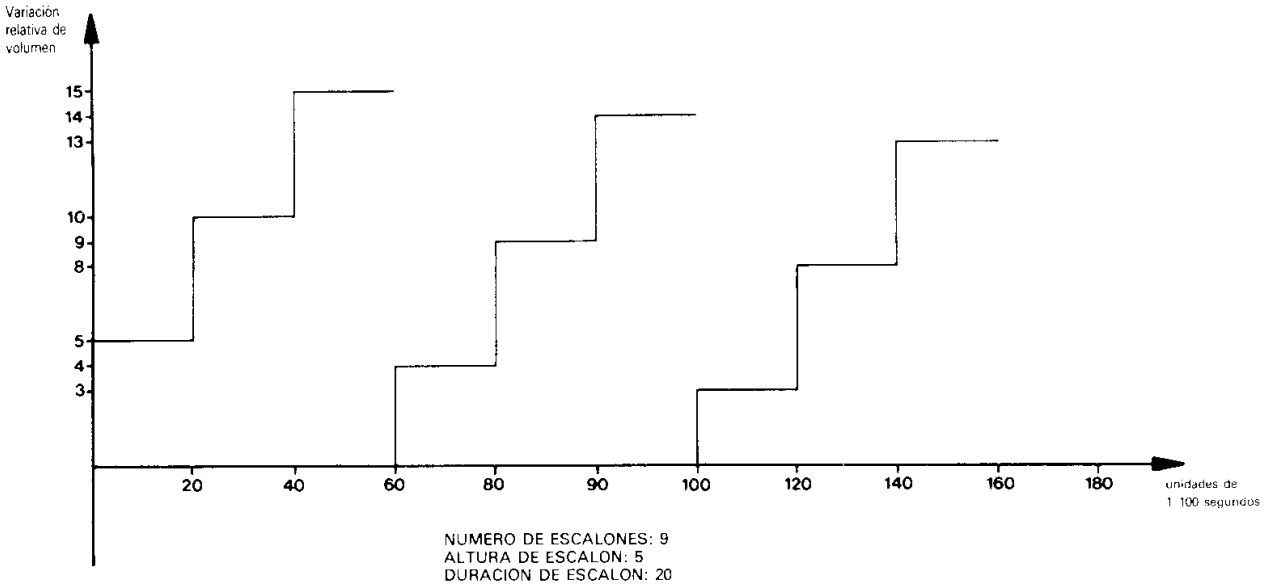
la envolvente consta de una sola sección de **9** etapas, cada una de las cuales incrementa el volumen en **5** unidades y dura **20** centésimas de segundo. Al terminar la tercera etapa, el volumen ha subido de 0 a 15; en la cuarta etapa volverá a 4; en la quinta a 9, etc. El proceso está ilustrado en la figura de la página siguiente.

El margen de valores para el número de escalones es de 0 a 127. En cada uno el valor puede variar entre −128 y +127 (los valores negativos representan disminución). La duración de cada escalón puede ser de entre 0 y 255 (centésimas de segundo).

Ya sabemos cómo variar el volumen de una nota. Ahora podemos estudiar cómo se define la variación de su tono con el tiempo para producir efectos tales como el 'vibrato'.

El método es muy parecido al de las envolventes de volumen. Las envolventes de tono se definen con la instrucción **ENT**. Por ejemplo:

```
ENT 1,5,1,1,5,-1,1
SOUND 1,284,10,15,,1
```



Las envolventes de tono se invocan con la instrucción **SOUND** poniendo en ésta como sexto parámetro el número de referencia de la envolvente. La instrucción **ENT** tiene que ser ejecutada antes que la **SOUND** que la utiliza.

Este primer ejemplo de **ENT** define la envolvente de tono número **1**. La primera sección consta de **5** etapas; en cada una el periodo de tono crece en **1** unidad; cada una dura **1** centésima de segundo. La segunda sección consta de **5** etapas; la variación del periodo de tono en cada una de ellas es **−1** (descenso); la duración de cada etapa es de **1** centésima de segundo. La duración total es, pues, 5+5=10 centésimas de segundo. Obsérvese que esta duración ha sido especificada en **SOUND**, ya que la duración de la envolvente de tono *no* determina la duración de la nota (mientras que sí lo hace la envolvente de volumen). Si la duración especificada en **SOUND** es menor que la de la envolvente, el final de ésta se perderá. Si es mayor, el final de la nota se ejecutará a tono constante. Esto último es también aplicable a las envolventes de volumen.

(Nótese la ausencia del número de envolvente de volumen en la última instrucción **SOUND**, debida a que aún no hemos definido una envolvente de volumen para este sonido.)

Las envolventes de tono normalmente duran menos que la nota. Se puede hacer que la envolvente se repita mientras la nota está sonando. Para ello se especifica un número de envolvente negativo, cuyo valor absoluto se cita en la instrucción **SOUND**:

```
ENT -5,4,1,1,4,-1,1
SOUND 1,284,100,12,,5
```

Esta repetición de la envolvente produce el efecto 'vibrato'. Cuando se diseñan envolventes de tono, es conveniente que el tono varíe simétricamente con respecto al valor inicial, de

forma que al repetir la envolvente el tono no se desvíe demasiado con respecto al valor central. Pruebe el siguiente sonido:

```
ENT -6,3,1,1
SOUND 1,284,90,12,,6
```

Habrá observado que la frecuencia ha disminuido drásticamente. Esto ha ocurrido porque la envolvente impone un aumento del periodo de tono de **3** unidades y se repite 30 veces (**90/3**). No obstante, ese efecto se puede aprovechar para imitar trinos y sirenas:

```
ENT -7,20,1,1,20-1,1
SOUND 1,100,400,12,,7

ENT -8,60,-1,1,60,1,1
SOUND 1,100,480,12,,8
```

Se pueden definir hasta 15 envolventes de tono, con números de referencia del 1 al 15; los números negativos indican que la envolvente se repite. Para cada sección, el número de escalones puede estar entre 0 y 239. La variación del periodo de tono en cada escalón puede ser de entre −128 y +127. La duración de cada escalón puede ser de entre 0 y 255 (centésimas de segundo). Cada envolvente puede tener hasta 5 secciones.

El último parámetro que se puede incluir en la instrucción **SOUND**, el séptimo, caracteriza el nivel de ruido que se añade al sonido. Obsérvese que sólo hay un canal de ruido y que, por consiguiente, cada vez que se especifica un nivel de ruido se anula la anterior especificación.

El ruido se puede mezclar con un tono, pero también se lo puede programar por separado, para lo cual se debe poner un **0** como periodo de tono en **SOUND**. Esto es útil para imitar ruidos de percusión:

```
ENT -3,2,1,1,2,-1,1
ENV 9,15,1,1,15,-1,1
FOR a=1 TO 10:SOUND 1,4000,0,0,9,3,15:NEXT
```

Un ruido como éste puede servir de base para imitar el de una locomotora. Obsérvese que hemos combinado los dos tipos de envolvente y el ruido. En **SOUND** hemos puesto **0** para los parámetros de duración y de volumen, por lo que estas características quedan controladas por las envolventes de volumen.

Como ya estamos en condiciones de utilizar **SOUND**, **ENV** y **ENT** a plena potencia, vamos a estudiar algunas otras órdenes y funciones.

Como el lector recordará, al describir el primer parámetro de **SOUND** dijimos que si le sumábamos el número 64 el sonido quedaba 'retenido' en la cola, y que no sonaría mientras no lo liberásemos. La forma de liberarlo es ejecutar la orden **RELEASE**. Esta palabra va seguida de un número cuyos bits determinan a qué canales afecta la orden:

4 significa canal C
2 significa canal B
1 significa canal A

Los canales se combinan sumando los números correspondientes. Así, para liberar el sonido de los tres canales la orden que se requiere es:

```
RELEASE 7
```

donde 7=1+2+4. Si no hay sonido retenido en ningún canal, la orden no tiene efecto. Pruebe lo siguiente:

```
SOUND 1+64,90
SOUND 2+64,140
SOUND 4+64,215
RELEASE 3:FOR t=1 TO 1000:NEXT:RELEASE 4
```

No se produce ningún sonido mientras no se ejecuta la primera orden **RELEASE**, la cual libera los sonidos de los canales A y B. Después de una pausa, la segunda orden **RELEASE** libera el canal C.

Hay otro método para sincronizar sonidos. Cuando se retiene un sonido sumando 64 a su número de situación de canal, no sólo queda él retenido, sino todos los que se envíen a continuación a ese mismo canal. Si se envían más de cuatro sonidos a una cola que está retenida, la máquina queda bloqueada hasta que se libere la cola (posiblemente al ejecutarse una subrutina invocada por **AFTER** o **EVERY**). Ésta no es una buena forma de gestionar los sonidos, ya que la máquina se detendrá cada vez que se llene una cola. Lo mismo ocurre si se ejecutan varias órdenes **SOUND** seguidas. Pruebe este programa:

```
10 FOR a=1 TO 8
20 SOUND 1,100*a,200
30 NEXT
40 PRINT"hola"
run
```

El texto no aparece en la pantalla inmediatamente, sino al cabo de tres segundos. Esto ocurre porque el programa no puede llegar a la línea 40 mientras no haya espacio libre en la cola para almacenar todos los sonidos.

BASIC dispone de un mecanismo de interrupción, similar al que se utiliza en **AFTER**, **EVERY** y **ON BREAK GOSUB**, mediante el cual se puede hacer que se ejecute una subrutina cada vez que queda espacio libre en una cola especificada:

```
10 a=0
20 ON SQ(1) GOSUB 1000
30 PRINT a;
40 GOTO 30
1000 a=a+10
1010 SOUND 1,a,200
1020 IF a<200 THEN ON SQ(1) GOSUB 1000
1030 RETURN
```

Observe que el programa no se detiene. La instrucción **SOUND** no se ejecuta mientras no hay espacio libre en la cola del canal A (**1**), hecho que detecta la orden **ON SQ(1) GOSUB** de la línea 20. Esta orden inicializa un mecanismo de interrupción que ejecuta la subrutina cada vez que queda un hueco libre en la cola especificada. El mecanismo tiene que ser reinicializado cada vez que se ejecuta la subrutina (línea 1020). En este ejemplo, la reinicialización sólo se produce si **a** es menor que 200.

En un programa que lleve a cabo acciones relativamente lentas (por ejemplo, mover objetos por la pantalla), se puede poner música de fondo programando una subrutina que ejecute una nota cada vez que quede un hueco libre en la cola. De esta forma se asegura que el programa no se va a detener en espera de que quede espacio en la cola. Si los valores de las notas están contenidos en listas **DATA**, se puede hacer que la subrutina de sonido deje de reinicializarse cuando los datos estén a punto de agotarse.

El parámetro que va entre paréntesis en la instrucción **ON SQ( ) GOSUB** puede ser 1, 2 o 4, dependiendo del canal cuya cola se desee examinar.

Hay una *función*, **SQ( )**, que se puede utilizar para determinar el estado de los canales de sonido. Su parámetro puede ser 1, 2 o 4. El valor generado por la función se interpreta bit a bit según la siguiente tabla:

Bit	Decimal	Significado
0, 1, 2	1 a 4	Número de huecos libres en la cola
3	8	La primera nota de la cola está marcada para sincronizar con el canal A
4	16	La primera nota de la cola está marcada para sincronizar con el canal B
5	32	La primera nota de la cola está marcada para sincronizar con el canal C
6	64	La primera nota de la cola está retenida (está a 1 el bit de retención)
7	128	En este momento está sonando una nota

Pruebe el siguiente ejemplo:

```
10 SOUND 2,200
20 x=SQ(2)
30 PRINT BIN$(x)
run
```

La línea 30 escribe el número binario **10000100**. El bit 7 está a 1, lo que indica que en el canal había una nota sonando cuando se ejecutó la línea 20. Los tres dígitos menos significativos son **100**; equivalen al número decimal 4, y esto quiere decir que había cuatro espacios libres en la cola. Esta función examina la situación del canal en un punto específico del programa; en cambio, **ON SQ( ) GOSUB** examina la cola, y reacciona en consecuencia, en un punto indeterminado.

Hasta ahora todos los ejemplos han consistido en hacer sonar una o dos notas. Si se va a ejecutar un grupo de notas independientes, por ejemplo las de una melodía, sus característi-

cas se pueden guardar en líneas **DATA**, para luego leerlas con **READ** e introducirlas en **SOUND**:

```
10 FOR octava=-1 TO 2
20 FOR x=1 TO 7:REM notas por octava
30 READ nota
40 SOUND 1,nota/2↑octava
50 NEXT
60 RESTORE
70 NEXT
80 DATA 426,379,358,319,284,253,239
run
```

El ejemplo final de esta sección se basa en este concepto. En los canales A y B se hace sonar una melodía con ritmo, utilizando la sincronización. Este ejemplo muestra cómo utilizar las listas **DATA** para incluir información sobre nota, octava, duración y sincronización:

```
10 REM la linea 190 da la melodia en clave de agudos
20 REM la linea 200 da la melodia en clave de graves
30 DIM escala%(12):FOR x%=1 TO 12:READ escala%(x%):NEXT
40 canal1%=1:READ canal1$:canal2%=2:READ canal2$
50 CLS
60 velocidad%=12
70 escala$=" a-b b c+c d-e e f+f g+g"
80 ENV 1,2,5,2,8,-1,10,10,0,15
90 ENV 2,2,7,2,12,-1,10,10,0,15
100 ENT -1,1,1,1,2,-1,1,1,1,1
110 DEF FNm$(s$,s)=MID$(s$,s,1)
120 canal1%=1:GOSUB 200
130 canal2%=1:GOSUB 380
140 IF canal1%+canal2%>0 THEN 140
150 END
160 DATA &777,&70c,&6a7,&647,&5ed,&598
170 DATA &547,&4fc,&4b4,&470,&431,&3f4
180 DATA 4cr4f4f1f1g1A1-B2C2f4g2g1A1-B6A2Cr1f1g1f1g1a1-b1A1-b2
    C2g2A2g2f1g1a2g2f6e2c2e2c2g2e2c1-B1A2g2f4e4d8c4f3f1c2d4-b2fr2
    -B2A2g2f6e2gr4C4-B1a1f1-b1g2c2-b4a4g4fr6A2A2-B4-B2Ar2-B2
    A2g2f6e2g4C4-B1A1f1-B1g2C2-B4A4g8f.
190 DATA r4f4f8f4e4c4fr8f4e2f2e4d2e2d8c8c6e2f4g4g8e4f3f1c4dr8g4cr4e4c6
    f2d4c4c8fr8-e4dr8g8c4e4c6f2d4c4c8f.
200 REM enviar sonido al canal A
210 p1$=FNm$(canal1$,canal1%)
220 IF p1$<>"r" THEN r1%=0:GOTO 240
230 r1%=16:canal1%=canal1%+1:p1$=FNm$(canal1$,canal1%)
240 IF p1$="." THEN canal1%=0:RETURN ELSE l1%=VAL(p1$)
```

```
250 canal1%=canal1%+1
260 n1$=FNm$(canal1$,canal1%)
270 canal1%=canal1%+1
280 IF n1$="+" OR n1$="-" THEN 350
290 n1$=" "+n1$
300 nd1%=(1+INSTR(escala$,LOWER$(n1$)))/2
310 IF ASC(RIGHT$(n1$,1))>96 THEN o1%=8 ELSE o1%=16
320 SOUND 1+r1%,escala%(nd1%)/o1%,velocidad%*l1%,0,1,1
330 ON SQ(1) GOSUB 200
340 RETURN
350 n1$=n1$+FNm$(canal1$,canal1%)
360 canal1%=canal1%+1
370 GOTO 300
380 REM enviar sonido al canal B
390 p2$=FNm$(canal2$,canal2%)
400 IF p2$<>"r" THEN r2%=0:GOTO 420
410 r2%=8:canal2%=canal2%+1:p2$=FNm$(canal2$,canal2%)
420 IF p2$="." THEN canal2%=0:RETURN ELSE l2%=VAL(p2$)
430 canal2%=canal2%+1
440 n2$=FNm$(canal2$,canal2%)
450 canal2%=canal2%+1
460 IF n2$="+" OR n2$="-" THEN 530
470 n2$=" "+n2$
480 nd2%=(1+INSTR(escala$,LOWER$(n2$)))/2
490 IF ASC(RIGHT$(n2$,1))>96 THEN o2%=4 ELSE o2%=8
500 SOUND 2+r2%,escala%(nd2%)/o2%,velocidad%*l2%,0,1,2
510 ON SQ(2) GOSUB 380
520 RETURN
530 n2$=n2$+FNm$(canal2$,canal2%)
540 canal2%=canal2%+1
550 GOTO 480
run
```

Para ampliar información ...

Hay numerosas publicaciones de Indescomp y de otras editoriales en las que puede encontrar más amplia información sobre el CPC464; en concreto, las especificaciones de BASIC y del firmware.

Finalmente vamos a ofrecerle tres apéndices: un glosario de terminología informática, los listados de unos cuantos programas y un índice alfabético.

Esperamos que haya encontrado este manual instructivo e interesante y le agradecemos que haya adquirido el CPC464.

Apéndice 1
Léxico inglés-castellano y glosario

Algunos términos que se utilizan frecuentemente en informática, explicados para los usuarios del CPC464.

Accumulator — **Acumulador**

Una parte del microprocesador, en el corazón del ordenador, en el que se almacenan temporalmente los datos mientras se los está procesando. Se lo utiliza en la programación en código de máquina. Los usuarios de BASIC ni siquiera necesitan saber que existe.

Acoustic coupler — **Acoplador acústico**

También denominado 'modem acústico'. Dispositivo electrónico mediante el cual se puede conectar el ordenador a la red telefónica. De esta forma el ordenador se puede integrar en una red de comunicación pública, o bien comunicarse con otros usuarios de ordenadores domésticos para el intercambio de información o programas.

Address — **Dirección**

Número que identifica la posición de una célula de la memoria del ordenador. Mediante la dirección se selecciona una célula concreta para leer su contenido o, en el caso de la memoria RAM, para leer o escribir un valor en ella.

Adventure game — **Juego de aventura**

Una pasión para algunos, un aburrimiento para otros. Un juego de ordenador en el que se invita al jugador a participar en una serie de sucesos pseudoaleatorios con el objetivo de salir de un laberinto.

Algorithm — **Algoritmo**

Nombre grandilocuente que se da a una fórmula o a una suma complicada. Secuencia de operaciones lógicas y aritméticas con las que se realiza una determinada tarea informática.

Alphanumeric — Alfanumérico

Adjetivo que describe la diferencia entre una letra o un dígito y un carácter gráfico.

ALU (Arithmetic and Logic Unit) — Unidad aritmética y lógica

Parte del microprocesador que realiza las operaciones lógicas y aritméticas. De interés solamente para la programación en código de máquina.

Ambiguous File Name — Nombre de fichero ambiguo

Un nombre de fichero que contiene uno o varios símbolos comodín. Los nombres de fichero ambiguos pueden representar al mismo tiempo varios ficheros cuyos nombres tienen algo en común.

AMSDOS

AMStrad Disc Operating System. Sistema operativo de disco AMSTRAD. Es el programa que permite que el BASIC de Locomotive acceda a los ficheros de disco.

AMSOFT

División de AMSTRAD especializada en el apoyo a los usuarios de los ordenadores; suministra programas, periféricos y publicaciones, específicamente con objeto de potenciar el CPC464 y sus muchas aplicaciones.

Analog — Analógico

Adjetivo que caracteriza los fenómenos en los que las transiciones entre el estado inicial y el final se realizan de forma gradual o continua. Los ordenadores son dispositivos digitales, mientras que los fenómenos naturales son en su mayoría analógicos; así pues, el ordenador tiene que realizar una conversión A/D antes de poder procesar los datos recibidos de una fuente analógica.

Animation — Animación

Los dibujos animados son el mejor ejemplo. La animación en los ordenadores consiste en mover gráficos por la pantalla, tratando de simular los movimientos de los seres reales.

Applications program — Programa de aplicaciones

Programa escrito para realizar una tarea específica, a diferencia de los programas de 'utilidad', tales como ensambladores, controladores de impresora, etc., que son de interés más general.

Arcade game — Juego gráfico de salón

Juego de ordenador basado en el movimiento de gráficos (invasores espaciales, monstruos al acecho en un laberinto, etc.). El usuario tiene que evitar la 'muerte' de los 'buenos'. Generalmente sirven para ejercitar los reflejos, pero no tienen ningún valor didáctico para los estudiantes de informática.

Architecture — Arquitectura

Término que designa la organización general de un ordenador (relaciones entre el bus de datos, la UCP y los periféricos).

Argument — Argumento

Una variable independiente. Por ejemplo, en **SIN(*x*)** la *x* es el argumento de la función **SIN**.

Array — Matriz

Combinación de elementos de un mismo tipo dispuestos en una o varias dimensiones. En el caso unidimensional se denominan 'vectores'.

Artificial intelligence AI — Inteligencia artificial

Técnica de programación que permite al programa aprender por la experiencia.

ASCII

American Standard Code for Information Interchange. Código estadounidense estándar para el intercambio de información. Forma habitual de representar dígitos, letras y otros símbolos mediante códigos numéricos.

Assembler — Ensamblador

Programa que facilita la programación en código de máquina. Las instrucciones son los llamados 'nemónicos': grupos de letras que se recuerdan e interpretan fácilmente porque dan una idea de la función que realiza cada instrucción.

Backup — Copia de seguridad

Copia que se hace de los programas o ficheros de datos en previsión de que se borre o deteriore accidentalmente el original.

Bar code — Código de barras

Código impreso legible por ordenador mediante técnicas ópticas tales como láser de baja potencia. (Ejemplo: el grupo de rayas verticales que se imprime en la cubierta de algunas revistas.)

Base — Base

Es el número de dígitos que se utilizan en cada sistema de representación numérica. En el sistema decimal es 10; en el binario, 2; en el hexadecimal, 16.

BASIC

Beginners' All-purpose Symbolic Instruction Code. Código de instrucciones simbólicas de uso general para principiantes. Es un lenguaje de programación interpretado que se incorpora en la mayor parte de los ordenadores domésticos. BASIC fue diseñado primordialmente como ayuda para la enseñanza, pues permite la comprobación de los programas en cualquier momento de su desarrollo. En esto se distingue de los programas compilados, que no pueden ser probados parcialmente mientras el programa no esté completo.

Baud — Baudio

Unidad de medida de la velocidad de transmisión de datos. Es un concepto distinto del de bit por segundo, aunque en muchos casos sus valores coinciden.

BCD (Binary Coded Decimal) — Decimal codificado en binario

Sistema de codificación para números decimales en el que cada dígito se representa por un grupo de cuatro bits.

BDOS

Basic Disc Operating System. Sistema operativo básico de disco. Parte del sistema operativo CP/M que proporciona un interfaz para que el programa del usuario pueda utilizar las funciones de CP/M.

Benchmark — Prueba patrón

Una tarea estándar que se realiza con distintos ordenadores para comparar su velocidad, precisión, etc. Por ejemplo, elevar al cuadrado el número 99999 y luego calcular la raíz cuadrada.

Binary — Binario

Sistema de numeración de base 2, en el que los números se representan como combinaciones de solamente dos dígitos: el 0 y el 1. (Véase la parte 1 del capítulo 'Cuando usted guste . . .'.)

Binary number — Número binario

Un número representado en el sistema binario. En el BASIC del CPC464 se los identifica por el prefijo **&X**; por ejemplo, **&X0101**=5(decimal).

BIOS

Basic Input/Output System. Sistema básico de entrada/salida. Es la parte de CP/M que se escribe específicamente para cada ordenador concreto. Gestiona todas las entradas y salidas: pantalla, teclado, disco, etc.

Bit

Forma abreviada de BInary digiT (dígito binario). Un dígito binario es uno de los dos símbolos (el 0 y el 1) con los que se representan los números en el sistema de base 2.

Bit significant — Interpretable bit a bit

Hace referencia a los casos en los que se puede extraer información de un número examinando el estado de cada uno de sus bits. El valor global en decimal no tiene significado evidente.

Boolean algebra — Algebra de Boole

Estructura algebraica sobre variables que pueden tomar dos valores distintos: verdadero/falso, 1/0, etc.

Boot — Carga inicial

Proceso de carga de un sistema operativo en la memoria del ordenador. Cuando se carga CP/M desde BASIC, lo primero que se lee es un pequeño programa autocargador, el cual realiza la carga del sistema operativo propiamente dicho.

Booting, Bootstrapping — Autocarga

Los programas y sistemas operativos no se cargan a sí mismos, sino gracias a una rutina (generalmente grabada en ROM) que inicia el proceso de carga a partir de una determinada posición de memoria.

Buffer — Tampón

Area de memoria reservada para el almacenamiento temporal en los procesos de transferencia de la información.

Bug — Error de programación

Un problema no previsto en un programa: puede ir desde una 'característica inesperada' (por ejemplo, si se pulsan cuatro teclas a un tiempo la pantalla cambia de color) hasta una secuencia de instrucciones que provoca la pérdida del control de la máquina.

Built-in commands — Órdenes internas o incorporadas

Órdenes que forman parte del sistema operativo. Son de ejecución más rápida que las órdenes externas, pues no tienen que ser cargadas del disco.

Bus

Grupo de conexiones, internas al ordenador o de entrada/salida, por las que se transmite información acerca del estado de la UCP, de la RAM o de cualquier otra parte del hardware. EL bus del CPC464 es accesible a través del zócalo **EXPANSION** que está en la cara posterior del ordenador.

Byte — Byte, octeto

Grupo de 8 bits. Es la menor porción de información que una UCP puede intercambiar con la memoria.

CAD, Computer Aided Design — Diseño asistido por ordenador

Generalmente, un programa interactivo que convierte el ordenador en un tablero de dibujo electrónico que además es capaz de realizar cálculos matemáticos.

Cartridge — Cartucho

Dispositivo de memoria externo compacto conectable al ordenador (v. g., programas en ROM, cuya ventaja es la gran velocidad de carga, pero que son más caros que los suministrados en disco).

Cassette

Cinta encapsulada, ya sea virgen o con programas comerciales.

CCP — Intérprete de consola

Console Command processor. Procesador de órdenes de consola. Es el módulo de CP/M que interpreta y ejecuta las entradas por el teclado.

Character — Carácter

Cualquier símbolo que pueda ser manejado y visualizado por el ordenador: dígitos, letras y otros signos, incluidos los símbolos gráficos.

Character set — Juego de caracteres

Conjunto de todos los signos y símbolos disponibles en un ordenador o una impresora. El hecho de que un ordenador tenga un símbolo determinado no implica que éste sea reproducible en la impresora.

Character string — Cadena literal

Grupo de caracteres que pueden ser almacenados y tratados en bloque, v. g. una palabra o grupo de palabras.

Chip — Pastilla

Término popular, aunque incorrectamente utilizado, para designar un circuito integrado monolítico. La 'pastilla' es en realidad un trozo de silicio en el que se graba el circuito.

Clock — Reloj

Sistema temporizador de referencia del ordenador que controla y sincroniza la secuencia de operaciones. Un reloj de tiempo real es el que mantiene la hora, fecha, etc.

Code — Código

Aparte de otros significados, forma abreviada de referirse al 'código de máquina'.

Cold start — Arranque en frío

Proceso de carga e inicialización de un sistema operativo. El arranque en frío de CP/M se realiza cuando se ejecuta la orden **|CPM** de AMSDOS.

Command — Orden

Instrucción de programa o instrucción directa de teclado.

Compiler — Compilador

Un programa complejo que convierte programas completos escritos en un lenguaje de alto nivel, tal como BASIC, en códigos directamente utilizables por el microprocesador, con el consiguiente incremento de la velocidad de ejecución.

Computer Aided Education — Enseñanza asistida por ordenador

Programa que facilita la enseñanza en general sirviéndose de los ordenadores.

Computer generations — Generaciones de ordenadores

Diversos hitos tecnológicos han delimitado las etapas de la evolución de la tecnología informática; los ordenadores fabricados con un nivel de sofisticación similar forman una 'generación'.

Console mode — Modo de consola

Modo directo de CP/M; en la pantalla aparece el mensaje inductor **A**〉 para indicar que el sistema está esperando que se teclee una orden interna o el nombre de un programa de ayuda.

Corruption — Corrupción

Destrucción o alteración del contenido de un fichero o de la memoria, indeseada y posiblemente irremediable.

CP/M

Control Program for Microcomputers. Programa de control para microordenadores. Sistema operativo basado en disco, desarrollado por Digital Research, que proporciona un interfaz con el que utilizar una gran variedad de programas.

CPU, Central Processing Unit — Unidad central de proceso

Núcleo del sistema informático que interpreta las instrucciones y hace que sean obedecidas. En los microordenadores la UCP es el propio microprocesador.

Cursor

Símbolo móvil que indica en qué lugar de la pantalla aparecerá el próximo carácter que se envíe a ella.

Cursor control keys — Teclas de movimiento del cursor

Teclas con las que se hace que el cursor se mueva por la pantalla. Frecuentemente se las utiliza también en los juegos gráficos.

Daisy-wheel printer — Impresora de margarita

Impresora que escribe con calidad de máquina de escribir. Los caracteres están preformados, cada uno en un brazo de la 'margarita', y se los transfiere al papel al golpear el brazo con un martillo.

Database — Base de datos

Colección de datos estructurados, integrados y coherentes, utilizables por el ordenador.

Data capture — Captación de datos

Recolección de datos enviados al ordenador desde fuentes externas.

Debugging — Depuración

Proceso de detección y eliminación de errores de los programas.

Decimal notation — Notación decimal

Sistema decimal de numeración, de base 10, que utiliza los dígitos del 0 al 9. Es el sistema ordinario.

Default — Prescrito, implícito, por defecto

Valor que el ordenador supone en ausencia de otra especificación por parte del usuario. Por ejemplo, en CP/M la unidad de disco implícita es la **A**:.

Delimiter

Véase Separator.

Diagnostic — Diagnóstico

Mensaje emitido automáticamente por el ordenador para señalar e identificar un error de un programa.

Digital

Es el término opuesto a 'analógico'. Describe la transición brusca entre dos estados de un proceso.

Digitiser — Digitalizador

Método de conversión de información analógica en digital.

Directory — Directorio

Sección de un disco que contiene las reseñas (títulos y características) de los ficheros grabados en él. Lista del contenido de un disco.

Disc, disk — Disco

Hoja de plástico circular, fina y plana, recubierta de material magnético, que se utiliza como medio de almacenamiento de la información. El disco está incluido en una envoltura protectora en la que hay una 'ventana' para la cabeza lectora. En el disco de 3 pulgadas la ventana está cubierta por un obturador metálico que se abre automáticamente cuando se introduce el disco en la unidad.

Disc drive — Unidad de disco

Mecanismo y circuitos electrónicos con los que se mueve el disco para leer o grabar datos en él.

Documentation — Documentación

Manuales suministrados con los ordenadores o los programas, en los que se explica su funcionamiento.

DOS (Disc Operating System) — Sistema operativo basado en disco

Programas que controlan todas las operaciones de la unidad de disco.

Dot matrix — Matriz de puntos

Retícula rectangular de puntos con la que se puede visualizar cada carácter seleccionando los puntos adecuados.

Double sided — De doble cara

Hace referencia a los discos en los que se puede almacenar información en ambas caras. Una unidad de doble cara puede acceder a ambas caras del disco sin necesidad de extraer y darle la vuelta a éste.

Dr. LOGO

Versión de LOGO elaborada por Digital Research. LOGO es un lenguaje de programación caracterizado por sus 'gráficos de tortuga'.

Dumb terminal — Terminal pasivo

Terminal de ordenador que actúa solamente como medio de entrada o salida, pero sin procesar la información.

Edit — Editar, corregir

Corregir o modificar datos, programas o textos.

Editor

Un programa, generalmente grabado en la ROM del ordenador, que permite realizar el proceso de edición.

EPROM

Erasable Programmable Read only Memory. Memoria borrable y programable de 'sólo lectura'. Similar a la PROM, pero tal que su contenido se puede borrar con luz ultravioleta para luego grabar en ella un nuevo programa. Una EEPROM puede ser borrada por medios electrónicos.

Expression — Expresión

Fórmula utilizada en un programa para realizar cierta operación con los datos. La propia expresión indica normalmente la naturaleza de los datos que puede manejar. En Dr. LOGO una expresión consiste en el nombre de un procedimiento seguido de las entradas requeridas.

Fifth generation computers — Ordenadores de la quinta generación

Grandes ordenadores que estarán disponibles en el futuro y tendrán, como característica distintiva, la capacidad de utilizar ampliamente las técnicas de la inteligencia artificial.

File — Fichero

Colección de datos, de estructura homogénea, generalmente grabada en cinta o disco.

File name — Nombre de fichero

En Dr. LOGO el nombre de un fichero puede estar formado por un máximo de 8 caracteres alfanuméricos. En CP/M se permiten otros tres caracteres, precedidos por un punto.

Firmware

Software contenido en ROM: un híbrido de software y hardware.

Fixed-point number — Número en coma fija

Un número que se representa, manipula y almacena con el punto decimal en una posición determinada.

Floating-point number — Número en coma flotante

Número que se representa, manipula y almacena con la coma decimal en la posición más conveniente (en forma de mantisa y exponente).

Floppy disc

Véase Disc.

Flowchart — Ordinograma

Representación esquemática de la evolución de las acciones de un programa.

Forth

Lenguaje de programación de alta velocidad de ejecución. Tanto la velocidad como la complejidad están a medio camino entre los lenguajes de alto nivel y el código de máquina. No es un lenguaje para principiantes.

Function key — Tecla de función

Tecla que tiene asignada una tarea específica, que puede ejecutar en lugar (o además) de la función grabada sobre ella.

Gate — Puerta lógica

Las puertas lógicas permiten el paso de datos cuando se cumplen ciertas condiciones. Hay varios tipos de puertas lógicas: OR, AND, XOR, etc.

Graphics — Gráficos

Dibujo de puntos, rectas, circunferencias, etc.; es decir, todo lo que aparece en la pantalla que no sean 'caracteres'.

Graphics character — Carácter gráfico

Símbolo asignado a un carácter que es en sí mismo un dibujo.

Graphics cursor — Cursor gráfico

Concepto similar al de 'cursor de texto', pero invisible (en el 464) y referido a la pantalla gráfica.

Graphics mode — Modo gráfico

Los ordenadores más antiguos tenían que ser puestos explícitamente en este modo para poder manejar gráficos. En cambio, los ordenadores personales modernos pueden mezclar textos con gráficos.

Graphics tablet — Tablero gráfico

Periférico que convierte un dibujo en una forma manipulable por el ordenador. Es una forma de conversión A/D (analógica a digital).

Handshaking — Intercambio de indicativos y señales de control

Secuencia de señales electrónicas que inicia, comprueba y sincroniza el intercambio de datos entre un ordendor y un periférico o entre dos ordenadores.

Hard copy — Copia en papel

Listado por impresora de un programa o un texto, o volcado en papel de la pantalla gráfica.

Hardware — Equipos, máquinas

Componentes electrónicos o mecánicos de un sistema informático: todo lo que tiene volumen y peso.

Hexadecimal (HEX)

Sistema numérico de base 16. Los números hexadecimales se identifican en el CPC464 por los prefijos **&** y **&H**. Por ejemplo, **&FF**=255 (decimal). (Véase la parte 1 del capítulo 'Cuando usted guste . . .'.)

Hex file — Fichero HEX

Representación ASCII de un fichero de orden externo o de código de máquina.

High-level language — Lenguaje de alto nivel

Lenguajes del tipo de BASIC, en los que las instrucciones son más próximas al lenguaje ordinario. Son más lentos que el código de máquina, pero mucho más sencillos de entender y programar.

IEEE-488

Uno de los interfaces estándar para la conexión de dispositivos externos a un ordenador. Similar al Centronics, pero no totalmente compatible con él.

Information technology — Tecnología de la información

Todo lo relativo a la utilización de los ordenadores en el proceso de la información y las comunicaciones: proceso de textos, transmisión de datos, videotexto, teletexto, etc.

Initialise — Inicializar

Poner en marcha un sistema. O dar valores concretos a variables antes de empezar la ejecución del núcleo central de un programa; por ejemplo, dimensionar matrices, declarar tipos de variables, etc.

Input — Entrada

Datos de todo tipo que entran en el ordenador procedentes del disco, del teclado, de una puerta RS-232, etc.

Instruction — Instrucción

Orden que manda al ordenador que realice alguna operación o tarea. Un conjunto de instrucciones forma un programa.

Instruction set — Juego de instrucciones

Conjunto de procesos lógicos y matemáticos fundamentales que puede realizar el microprocesador. Toda instrucción de un lenguaje de alto nivel (y también los nemónicos de ensamblador) tiene que ser convertible a una instrucción o serie de instrucciones que la UCP pueda reconocer.

Integer — Entero

Número sin parte decimal.

Integrated circuit — Circuito integrado

Conjunto de componentes electrónicos miniaturizados, grabados en una pastilla de silicio. (Véase también Chip.)

Intelligent terminal — Terminal inteligente

Un terminal que, además de gestionar entradas y salidas para un ordendor central, puede realizar ciertas tareas de proceso de datos cuando el ordenador está 'fuera de línea'.

Interactive — Interactivo

Hace referencia a los programas que solicitan del usuario la introducción de información de modo inmediato: desde el control de naves espaciales en juegos gráficos hasta las respuestas en programas educativos. La acción del usuario tiene efecto en 'tiempo real' sobre la evolución del programa.

Interface — Interfaz

Vía de entrada y salida del ordenador, tanto para las señales eléctricas como para las acciones y reacciones del usuario. En el CPC464, el teclado, la pantalla y las conexiones para periféricos.

Interpreter — Intérprete

Una analogía más entre las instrucciones de ordenador y los idiomas. Es el elemento del sistema que interpreta el lenguaje de alto nivel y lo convierte a una forma utilizable por UCP de modo inmediato.

I/O — Entrada/Salida (E/S)

Input/output. Entrada/Salida.

Iteration — Iteración

Uno de los mecanismos básicos de la informática. Consiste en la repetición de tareas (bucles), en las que generalmente se va modificando el valor de una variable, hasta que se cumple determinada condición.

Joystick — Bastoncillo de mando

Dispositivo de entrada al ordenador que generalmente sustituye las teclas del cursor y hace que los juegos gráficos sean más fáciles y rápidos de controlar.

K

Forma abreviada y aproximada del prefijo 'kilo' (1000). Representa un 'kilobyte', que en realidad es 1024 bytes. (1024 es la base del sistema binario, 2, elevada a la décima potencia.)

Keyboard — Teclado

Matriz de interruptores alfanuméricos a través de la cual se introduce información en el ordenador.

Keyword — Palabra clave

Palabra que un programa o lenguaje de programación reserva para una función u orden específica.

Least significant bit — Bit menos significativo

En un número binario, el bit de menor peso; es decir, el que está más a la derecha en la representación matemática habitual.

Light pen — Lápiz óptico

Otro dispositivo de entrada al ordenador que sirve para indicar una selección de zonas de la pantalla o para leer códigos de barras.

Line number — Número de línea

Los programas de BASIC y de algunos otros lenguajes tienen las instrucciones numeradas para indicar al ordenador el orden de ejecución.

LISP

Acrónimo de 'LISt Processor language' (lenguaje de proceso de cadenas de predicados). Otro lenguaje de alto nivel.

Logic — Lógica

Entre otras acepciones, conjunto de componentes electrónicos que realizan las operaciones lógicas elementales en las que en último término se basa el funcionamiento del ordenador.

Logical device — Dispositivo lógico

La representación de un dispositivo puede no coincidir con su forma física. Por ejemplo, el dispositivo lógico **LST** de CP/M puede ser asignado a la puerta Centronics, pero también a la pantalla del monitor.

LOGO

Nombre de un lenguaje de programación tomado del término griego 'logos', que significa 'palabra'. LOGO ha sido diseñado para servir como ayuda en la enseñanza de la informática elemental.

Loop — Bucle

Sección de un programa que se ejecuta reiteradamente hasta que se cumple determinada condición.

Low-level languaje — Lenguaje de bajo nivel

Por ejemplo, el lenguaje ensamblador. Un lenguaje en el que cada instrucción corresponde a una instrucción de código de máquina.

LSB, Least Significant Bit — Bit menos significativo, bms

En un número binario, el bit de menor peso, es decir, el que está más a la derecha en la representación matemática habitual.

LSI, Large Scale Integration — Integración a gran escala

Etapa avanzada del desarrollo de circuitos integrados, en la que incorporan miles de componentes electrónicos en una sola pastilla de silicio.

Machine code — Código de máquina

Lenguaje de programación que es directamente comprensible y utilizable por el microprocesador.

Machine readable — Accesible a la máquina

Este término hace referencia a datos e información que puedan ser introducidos directamente en la máquina, sin necesidad de conversión.

Man-machine interface — Interfaz hombre-máquina

Dispositivo que facilita la interacción entre el usuario y el ordenador (teclado, sonido, etc.)

Matrix — Matriz

Disposición de los puntos con los que se forman los caracteres en la pantalla o en las impresoras de 'matriz de puntos'. En otra acepción, variables que con un solo nombre pueden contener múltiples valores, a los que se accede mediante los 'índices'.

Memory — Memoria

Especie de 'aparcamiento' en la máquina, en la que ésta almacena los programas y los datos. La memoria puede ser RAM (memoria de acceso aleatorio), en la que se puede escribir y leer, o ROM (memoria de 'sólo lectura'), en la que solamente se puede leer.

Memory map — Mapa de memoria

Esquema en que se indica qué zonas de memoria están dedicadas a las diversas funciones: pantalla, sistema operativo, variables, etc.

Menu — Menú

Lista de opciones que se ofrece al usuario de un programa.

Microprocessor — Microprocesador

Circuito integrado que controla el funcionamiento del ordenador. Recibe y ejecuta las instrucciones que le entrega el intérprete de BASIC y controla todas las interfaces.

Modem, Modulator/Demodulator — Modulador/demodulador, modem

Dispositivo que permite conectar el ordenador a una línea telefónica o comunicarlo con otros ordenadores o terminales remotos.

Monitor

Pantalla en la que se visualizan los gráficos, listados, mensajes, etc. También el programa que permite observar la operación básica del ordenador.

Mouse — Ratón

Seguidor boca abajo, o bola rodante. Al desplazarlo con la mano sobre una superficie plana, mueve el cursor por la pantalla.

MSB, Most Significant Bit — Bit más significativo, BMS

En un número binario, el bit de mayor peso; es decir, el que está más a la izquierda.

Network — Red

Cuando se interconectan varios ordenadores para intercambiar programas y datos, vía modem, cable, etc., se forma una 'red'.

Nibble — Cuarteto

Medio byte, es decir, un número binario de cuatro bits.

Node — Nodo

Unidad de espacio de trabajo en LOGO. Generalmente un nodo consume 4 bytes de espacio de memoria.

Noise — Ruido

El sistema de sonido del CPC464 permite generar ruidos aleatorios para simular efectos sonoros tales como explosiones, disparos, etc.

Numeric keypad — Teclado numérico

Grupo de teclas numéricas destinadas a facilitar la introducción de números en el ordenador. En el caso del CPC464, estas teclas son además definibles por el usuario.

OCR, Optical Character Recognition — Lectura óptica de caracteres

Sistema de lectura automática de caracteres escritos y de conversión a datos utilizables por el ordenador.

Octal

Sistema de numeración de base 8 en que cada dígito (del 0 al 7) se presenta con tres dígitos binarios.

Off line — Fuera de línea

Situación en la que se encuentra un periférico cuando no está activamente conectado al ordenador.

On line — En línea

Término opuesto a Off line/Fuera de línea.

Operating system — Sistema operativo

Programa que gestiona el manejo de la memoria, la ejecución de programas, prioridades, control de tiempos, etc. Interfaz entre la máquina y los programas del usuario.

Operator — Operador

Parte de una operación aritmética o lógica que describe en qué consiste la operación.

Output — Salida

Todo lo que sale del ordenador como resultado de algún proceso o cálculo.

Overwrite — Sobreescribir

Escribir borrando lo anterior. Borrar parte de la memoria o la pantalla, a veces involuntariamente, al escribir nuevos datos en ella.

Paddle — Empuñadura

Mando principal del joystick.

Page zero — Página cero

Zona de memoria, en CP/M entre las posiciones &0000 y &0100, en la que se almacenan parámetros esenciales del sistema operativo.

Parallel interface — Interfaz paralelo

Sistema de conexión del que es ejemplo el interfaz para impresora del CPC464. Cada línea del bus de datos se conecta a la correspondiente patilla del corrector de la impresora. Los datos se transmiten en grupos simultáneos. Este método de transmisión es mucho más rápido que el que se consigue con un interfaz serie, en que los dígitos binarios se envían uno tras otro, por una sola línea.

Pascal

Lenguaje estructurado de alto nivel que ha de ser compilado antes de su ejecución y que, por consiguiente, es muy rápido. Es el lenguaje que habitualmente se estudia después de BASIC.

PEEK

Función de BASIC que examina una posición de memoria especificada e informa de su contenido.

Peripheral — Periférico

Impresoras, modems, joysticks, magnetófonos, etc.; cualquier aparato que pueda ser conectado al ordenador para hacerlo más útil.

Physical device — Dispositivo físico

Dispositivo que realmente existe (o sea, pieza de hardware), representable por un dispositivo lógico.

Pixel (picture element) — Punto gráfico

La menor área de la pantalla que puede ser controlada con independencia de las demás.

Plotter — Trazador automático

Trazador especial que dibuja con plumas, tanto caracteres como dibujos propiamente dichos.

POKE

Orden de BASIC que escribe un valor en una posición de memoria especificada.

Port — Puerta

Lugar específicamente direccionable de un interfaz para la entrada o salida de datos.

Portability — Adaptabilidad

Característica de un lenguaje o programa que hace fácil su adaptación a otra máquina, generalmente como consecuencia de la compatibilidad del sistema operativo (v. g., el CP/M de Digital Research).

Primitives — Primitivas

Procedimientos, operaciones u órdenes básicos en los que consiste Dr. LOGO.

Printer — Impresora

Máquina que escribe en papel los datos enviados por el ordenador.

Procedure — Procedimiento

Serie de expresiones o instrucciones que indican al ordenador cómo realizar una tarea.

Program — Programa

Conjunto de instrucciones que gobiernan el funcionamiento del ordenador para hacer que realice una tarea. Puede tratarse de una simple rutina en código de máquina o de una aplicación completa, tal como un paquete de contabilidad o un procesador de textos.

Programming language — Lenguaje de programación

Lenguaje en el que escribe un programa, consistente en un conjunto de reglas que indican cómo se utilizan las palabras y los números.

PROM

Programmable Read Only Memory. Memoria programable de 'sólo lectura'. Circuito integrado de memoria que, una vez grabado, ya no se puede borrar. (Véase también EPROM.)

Prompt — Mensaje inductor

Un corto mensaje o grupo de caracteres que recuerda al lector que el ordenador está esperando una entrada. Por ejemplo, en CP/M es >; en LOGO, ?.

PSU, Power Supply Unit — Fuente de alimentación

Circuito que convierte la tensión alterna de la red a una tensión continua adecuada para los circuitos del ordenador (y de los periféricos).

QWERTY keyboard — Teclado tipo QWERTY

Teclado del tipo de las máquinas de escribir estándar.

RAM

Random Access Memory. Memoria de acceso aleatorio. Memoria de la que se puede leer y en la que se puede escribir en el curso normal de la ejecución de un programa, en cualquier dirección y en cualquier orden.

Random access — Acceso aleatorio

Término que hace referencia a la posibilidad de leer o escribir datos en un disco en el orden que se desee.

Random number — Número aleatorio

Número generado por el ordenador que no es ni repetible ni predecible. El CPC464 genera una sucesión de números pseudoaleatorios.

Raster scan — Barrido de trama

Sistema de 'escritura' en la pantalla, en que las imágenes se forman mediante un barrido de líneas horizontales.

R/O, Read only — De 'sólo lectura'

Atributo de un disco, de un fichero, de una unidad de disco, etc., que impide la escritura y la modificación de los datos.

R/W, Read write — De lectura/escritura

Atributo de un disco, de un fichero o de una unidad de disco que permite tanto la lectura como la escritura o la modificación de los datos.

Real number — Número real

Número que consta (o puede constar) de parte entera y parte decimal.

Real time — En tiempo real

Efectos que ocurren en el momento presente, a diferencia de aquéllos que sólo se aprecian cuando ha terminado el proceso que los provocó.

Record — Registro

Unidad de medida de ficheros. En CP/M los registros son de 128 bytes.

Recursion — Recursividad

Serie de etapas repetidas de un programa o una rutina, tales que cada una está relacionada con la anterior.

Refresh — Renovación, refresco

Actualización de la información, ya sea de la pantalla o de la memoria. No siempre es un proceso destructivo; puede tratarse de una confirmación de la situación actual, en dispositivos que de otro modo perderían la información.

Register — Registro de máquina

Posición de memoria transitoria de la UCP utilizada para almacenamiento provisional de los datos.

REM, Remark — Comentario, observación

Instrucción de un programa que el ordenador no ejecuta; sirve para que el programador escriba anotaciones, o para escribir la fecha o número de versión del programa.

Reserved word — Palabra reservada

Una palabra que tiene un significado especial para un programa, y que por consiguiente no se puede utilizar fuera del contexto predefinido. Por ejemplo, BASIC no puede aceptar **NEW** como nombre de una variable, pues esa palabra está reservada para otro propósito.

Resolution — Resolución

La resolución es tanto mayor cuanto menor sea el tamaño de los pixels. Da una idea de cómo de 'finos' pueden ser los gráficos en un ordenador.

Reverse Polish notation — Notación polaca inversa

Método de escribir operaciones aritméticas, adoptado por ciertos fabricantes de calculadoras, en el que los operadores (+, −, *, /) se ponen después de los operandos.

RF Modulator — Modulador de radiofrecuencia

Método por el cual las señales de video del ordenador se codifican y transmiten a la entrada de antena de un televisor.

ROM

Read only Memory. Memoria de 'sólo lectura'. Memoria que no se puede borrar.

Routine — Rutina

Una parte del programa que realiza una tarea específica. Un subprograma contenido en un programa más general, o que se ha escrito de forma independiente y puede ser incorporado a otros programas.

RS-232C

Un sistema concreto estandarizado para los interfaces de comunicación en serie. Los dispositivos que están en los dos extremos de la línea deben estar configurados de acuerdo con las normas EIA RS-232C.

Screen editor — Editor de pantalla

Editor de textos o de programas que permite llevar el cursor a cualquier lugar de la pantalla para modificar su contenido.

Scrolling — Desplazamiento vertical, rodadura

Término que describe la forma en que la pantalla 'rueda' hacia arriba cuando se ha llegado a la última línea y se necesita una nueva.

Sector

Memoria mínima direccionable en un disco. El sistema de AMSTRAD está basado en sectores de 512 bytes.

Separator — Separador

Signo que marca el límite entre palabras reservadas y otros elementos del programa.

Serial interface — Interfaz serie

Este término hace referencia normalmente al estándar RS-232C, aunque existen otros métodos de transmisión en serie de datos.

Simulation — Simulación

Técnica de emulación de los procesos reales en un ordenador; por ejemplo, simulación de vuelo, de la conducción de automóviles, etc.

Singled sided — De una sola cara

Hace referencia al tipo de disco en el que sólo se pueden grabar datos en una cara.

Software — Programación, software

Conjunto de programas que hacen que el ordenador funcione; en disco, cinta, ROM, etcétera.

Software engineering — Ingeniería de programación

Técnicas de programación consideradas desde un punto de vista científico, para distinguirlas de las no organizadas o arbitrarias.

Sound generator — Generador de sonido

Parte de un ordenador (circuitos o programas) que genera sonido y ruido.

Speech recognition — Reconocimiento de voz

Conversión de la palabra hablada en información reconocible por el ordenador.

Speech synthesis — Síntesis de voz

Generación de voces simuladas mediante una combinación de circuitos y programas.

Spread sheet — Hoja electrónica

Un programa que permite introducir datos numéricos para disponerlos en filas y columnas y para hacer con ellos manipulaciones aritméticas. Al modificar uno de los datos se rehacen los cálculos para actualizar los resultados por filas y columnas.

Sprite — Agente gráfico móvil

Objeto gráfico que se puede mover libremente por la pantalla, generado por software o por hardware de forma tal que pueda aparecer y desaparecer de modo aparentemente aleatorio.

Stack — Pila

Área de memoria utilizada para almacenar datos provisionalmente y organizada de forma que el último dato depositado en ella es el primero que se recupera.

Stream — Canal

Vía de salida del ordenador; por ejemplo, la pantalla, la impresora o el disco.

String — Cadena literal

Un objeto formado por caracteres cualesquiera que el programa no puede tratar como numérico. Puede constar solamente de dígitos, pero el ordenador no lo considera como número mientras no se lo convierta a número mediante la instrucción adecuada.

Structured programming — Programación estructurada

Técnica de programación lógica y premeditada con la que se escriben programas que fluyen de principio a fin en etapas bien diferenciadas.

Sub-routine

Véase Routine, rutina.

Syntax error — Error de sintaxis

Cuando se infringen las reglas sobre la utilización de palabras clave y variables, BASIC emite este mensaje.

System tracks — Pistas del sistema

Pistas reservadas en el disco para el sistema CP/M.

Terminal

Teclado de entrada con pantalla, o bien con teletipo de salida.

TPA, Transient Program Area — Area para programas transitorios

Area de memoria que empieza en &0100, en la que en CP/M se almacena los programas de usuario y los datos por éstos generados.

Track — Pista

Las pistas son coronas circulares del disco. Cada una se divide en un número fijo de sectores. Las pistas y sectores se inscriben en áreas específicas del disco en el proceso de inicialización.

Transient program — Programa transitorio

Un programa de ayuda de CP/M, tal como **PIP**, que puede ser cargado en la TPA sin más que escribir su nombre en el teclado.

Truncated — Truncado

Hace referencia a un número o una cadena que han sido recortados por la izquierda o por la derecha. Cuando el truncamiento es intencionado, suele incluir el redondeo del número. Si es intencionado, los caracteres sobrantes se eliminan para encajar el número o la cadena en el espacio disponible.

Truth table — Tabla de verdad

El resultado de toda operación lógica puede ser 'verdadero' o 'falso' (para el ordenador, 1 o 0). La tabla de verdad da todos los resultados posibles de la operación en función del valor de los operandos.

Turtle — Tortuga

Un símbolo con forma de punta de flecha que funciona como cursor gráfico en la pantalla gráfica de Dr. LOGO.

Turtle graphics — Gráficos de tortuga

Imagen que deja la tortuga de Dr. LOGO al moverse por la pantalla.

UDK, User Defined Keys — Teclas definidas por el usuario

En el CPC464 hay 32 teclas a las que se pueden asignar tareas concretas, que el ordenador realiza en cuanto el usuario las pulsa.

Unsigned number — Número entero sin signo

Número entero interpretado como positivo.

Utility — Programa de ayuda

Un programa completo con el que se realiza una operación común a varias aplicaciones; por ejemplo, ordenar alfabéticamente o copiar ficheros.

Variable

Un dato o grupo de datos puede ser identificado en un programa mediante un nombre; su valor puede variar como consecuencia de la ejecución del programa.

Warm start — Arranque en caliente

Es el que se realiza cuando se pulsa [CTRL]C en CP/M. Reinicializa el subsistema de disco y devuelve el control al modo directo de consola de CP/M.

Wildcard character — Símbolo comodín

Este término hace referencia a los símbolos * y ?. Dr. LOGO solamente admite el ?. El * representa cierto número de símbolos ?. Al mencionar nombres de ficheros, se pueden construir nombres ambiguos poniendo símbolos ? en lugar de caracteres.

Write protection — Protección contra escritura

Un mecanismo que impide escribir en un disco o fichero. Un disco protegido contra escritura es un disco de 'sólo lectura'.

Apéndice 2
Que usted lo disfrute ...

Rebotes

Sencillo y apasionante. Para un jugador contra el ordenador. Teclado o joystick.

```
10 'REBOTES, por Alexander Martin
20 'copyright (c) AMSOFT 1984
30 '
40 MODE 1:BORDER 1:INK 0,1:INK 1,26:INK 2,24:INK 3,6
50 SPEED KEY 15,2
60 ENV 1,1,18,0,11,0,10
70 ENT 1,10,2,2
80 ENV 3,1,0,16,5,-3,2
90 ENV 2,5,3,3,1,-21,22,9,-3,2
100 ENT -2,10,2,2,5,-7,1,2,11,3,2,-4,8
110 '
120 '
130 MOVE 30,32:DRAWR 0,400,1:MOVE 610,32:DRAWR 0,400,1
140 PEN 3:LOCATE 3,1:PRINT STRING$(36,143)
150 PEN 2:LOCATE 3,2:PRINT STRING$(36,143)
160 PEN 1:FOR r=5 TO 6:LOCATE 3,r:PRINT STRING$(36,143):NEXT
    r
170 bx=9
180 pelotas=5:puntos=0
190 PEN 1:GOSUB 680
200 IF INKEY$<>CHR$(32) AND JOY(0)<16 THEN 200
210 LOCATE 3,23:PRINT SPACE$(36):LOCATE 1,24:PRINT SPACE$(40)
    ;
220 GOSUB 690:GOSUB 660:GOTO 280
230 '
240 '
250 LOCATE bx,24:PRINT"  ";STRING$(4,131);"  ":RETURN
260 '
270 '
280 xa=1:ya=1:IF INT(RND*2)=1 THEN xa=-xa
290 PEN 1:GOSUB 250
300 ORIGIN 0,400
310 x=bx+4:y=11:x1=x:y1=y
320 '
330 '
340 x1=x+xa:y1=y+ya
```

```
350 IF x1=3 OR x1=38 THEN xa=-xa
360 GOSUB 540
370 IF y1=24 AND x1>bx+1 AND x1<bx+6 THEN ya=-ya:y1=y1-2:SOUN
    D 130,44,8,7,1,1:a=((x>bx+5)OR(x<bx+2)):IF a=-1 THEN xa=x
    a*a:x1=x1+xa:y1=y1+1
380 IF y=25 THEN LOCATE x,y:PRINT" ":GOTO 500
390 GOSUB 250
400 t=TEST((16*x1)-1,-(16*y1)-1)
410 IF t<>0 THEN ya=-ya:xz=x1:yz=y1:y1=y1+ya:GOSUB 590:IF t=2
     THEN puntos=puntos+10:GOSUB 660
420 IF t=3 THEN puntos=puntos+20:GOSUB 660
430 IF t=1 THEN puntos=puntos+5:GOSUB 660
440 IF y1=1 THEN ya=1
450 LOCATE x,y:PRINT" ":LOCATE x1,y1:PRINT CHR$(231):x=x1:y=y
    1
460 IF y=1 OR x=3 OR x=38 THEN SOUND 129,78,8,7,1,1
470 GOTO 340
480 '
490 '
500 pelotas=pelotas-1:SOUND 132,19,46,12,2,2:IF pelotas=0 THE
    N GOTO 620
510 GOSUB 660:GOTO 280
520 '
530 '
540 IF (INKEY(8)=0 OR INKEY(74)=0) AND bx>2 THEN bx=bx-2:RETU
    RN
550 IF (INKEY(1)=0 OR INKEY(75)=0) AND bx<32 THEN bx=bx+2:RET
    URN
560 RETURN
570 '
580 '
590 LOCATE xz,yz:PRINT" ":RETURN
600 '
610 '
620 IF puntos>=maximo THEN maximo=puntos
630 GOSUB 660:puntos=0:pelotas=5:GOTO 130
640 '
650 '
660 SOUND 130,0,20,13,3,0,31:LOCATE 1,25:PRINT "Maximo";maxim
    o;
670 LOCATE 15,25:PRINT "Puntos";puntos:LOCATE 30,25:PRINT"Pel
    otas";pelotas:RETURN
680 LOCATE 3,23:PRINT"Pulse barra espaciadora para empezar":R
    ETURN
690 LOCATE 1,25:PRINT SPACE$(40);:RETURN
```

Bombardero

Una variación sobre un tema clásico. Para un jugador contra el ordenador. Sólo teclado.

```
10 'BOMBARDERO, por Dave Town
20 'copyright (c) AMSOFT 1984
30 '
40 MODE 1:CLS:INK 0,0:BORDER 0:INK 1,18:INK 2,6:INK 3,4:INK 5
   ,15:INK 6,2:INK 7,24:INK 8,8:INK 9,26:INK 10,10:INK 11,20:
   INK 12,12
50 SYMBOL AFTER 240:SYMBOL 241,&40,&60,&70,&7F,&7F,&EF,&7,&0:
   SYMBOL 242,&0,&32,&7A,&FE,&FA,&F2,&E0,&0
60 puntos=0:maximo=0:avi$=CHR$(241)+CHR$(242):x=2:y=2:cae=0:a
   =2:b=2
70 GOSUB 480
80 CLS
90 PEN 2:LOCATE 1,15:INPUT"Elija nivel: 0 (AS) a 5 (PRINCIPIA
   NTE) ",nivel
100 IF nivel<0 OR nivel>5 THEN GOTO 90
110 nivel=nivel+10
120 LOCATE 1,15:PRINT CHR$(18);:LOCATE 1,15:INPUT"Elija veloc
    idad: 0 (MAX) a 100 (MIN) ",vel
130 IF vel>100 OR vel<0 GOTO 120
140 '
150 'Edificios
160 '
170 MODE 0:FOR base=5 TO 15:FOR altura=21 TO INT(RND(1)*8+niv
    el) STEP-1:LOCATE base,altura:PEN base-2:PRINT CHR$(143)+
    CHR$(8)+CHR$(11)+CHR$(244);:NEXT:NEXT
180 PLOT 0,20,4:DRAW 640,20,4
190 LOCATE 1,25:PEN 2:PRINT"PUNTOS";puntos;:LOCATE 12,25: PRI
    NT"MAX";maximo;
200 '
210 'Juego
220 '
230 LOCATE x-1,y:PRINT"   ";
240 PEN 1:LOCATE x,y:PRINT avi$;:PEN 2
250 IF y=21 AND x=15 THEN GOTO 290:ELSE GOTO 340
260 '
270 'Aterriza
280 '
290 FOR c=0 TO 1000:NEXT
300 puntos=puntos+100-(nivel*2):nivel=nivel-1:x=2:y=2:a=2:b=2
    :cae=0
```

```
310 IF nivel<10 THEN nivel=10:vel=vel-20
320 IF vel<0 THEN vel=0
330 GOTO 150
340 FOR c=0 TO vel:NEXT
350 x=x+1
360 IF x=18 THEN LOCATE x-1,y:PRINT CHR$(18);:x=2:y=y+1:LOCAT
    E x,y:PEN 1:PRINT avi$;:PEN 2
370 a$=INKEY$:IF a$=" " AND cae=0 THEN cae=1:b=y+2:a=x
380 IF y=21 THEN cae=0
390 IF cae=1 THEN LOCATE a,b:PRINT CHR$(252);:LOCATE a,b-1:PR
    INT" ";:b=b+1:IF b>21 THEN LOCATE a,b:PRINT" ";:LOCATE a,
    b-1:PRINT" ";:a=0:b=0:cae=0:SOUND 3,4000,10,12,0,0,10
400 ga=(a-0.5)*32:gb=400-(b*16):bomba=TEST(ga,gb)
410 IF bomba>0 THEN GOTO 670
420 gx=((x+1.5)*32):gy=408-(y*16):choque=TEST(gx,gy)
430 IF choque>0 THEN GOTO 570
440 GOTO 230
450 '
460 'Instrucciones
470 '
480 LOCATE 1,2:PEN 1:PRINT"Usted esta pilotando un avion sobr
    e una ciudad desierta y tiene que pasar sobre los edifici
    os para aterrizar y repos-   tar. Su avion se mueve de iz
    quierda a   derecha.";:PRINT
490 PRINT:PRINT"Al llegar a la derecha, el avion vuelve a sal
    ir por la izquierda, pero MAS BAJO.Dispone de un numero l
    imitado de bombas y puede hacerlas caer sobre los edifi-
     cios pulsando la BARRA ESPACIADORA.";:PRINT
500 PRINT:PRINT"Cada vez que aterriza, sube la altura   de lo
    s edificios y la velocidad.";:PRINT:PRINT:PRINT"UNA VEZ D
    ISPARADA UNA BOMBA, YA NO PUEDE DISPARAR OTRA MIENTRAS NO
     HAYA EXPLOSIO-NADO LA PRIMERA!!!!";
510 PEN 2:LOCATE 1,24:PRINT:PRINT"Pulse una tecla para empeza
    r.";
520 a$=INKEY$:IF a$="" GOTO 520
530 RETURN
540 '
550 'Colision
560 '
570 LOCATE x-1,y:PRINT CHR$(32)+CHR$(32)+CHR$(32)+CHR$(253)+C
    HR$(8)+CHR$(238)+CHR$(8);
580 FOR t=1 TO 10:SOUND 7,4000,5,15,0,0,5:PEN t:PRINT CHR$(25
    3)+CHR$(8)+CHR$(238)+CHR$(8)+CHR$(32)+CHR$(8);:FOR tm=0 T
    O 50:NEXT:NEXT:PEN 2
590 CLS:LOCATE 1,5:PRINT"Ha conseguido";puntos;"puntos."
```

```
600 IF puntos>maximo THEN maximo=puntos:LOCATE 1,8:PRINT"BATI
    O EL RECORD!!";
610 puntos=0:LOCATE 1,12:PRINT"Pulse V para volver a empezar"
    ;
620 a$=INKEY$:IF a$="v" OR a$="V" GOTO 630 ELSE GOTO 620
630 PEN 1:MODE 1:x=2:y=2:a=2:b=2:GOTO 90
640 '
650 'Edificio bombardeado
660 '
670 LOCATE a,b-1:PRINT" "+CHR$(8);:PEN 4:FOR tr=1 TO INT(RND(
    1)*3)+1:puntos=puntos+5:SOUND 3,4000,10,12,0,0,10:LOCATE
    a,b:FOR t=0 TO 4:PRINT CHR$(253)+CHR$(8)+CHR$(32)+CHR$(8)
    ;:NEXT:b=b+1
680 IF b=24 THEN b=b-1
690 NEXT
700 LOCATE 7,25:PRINT puntos;:cae=0:a=x:b=y:GOTO 230
```

Amthello

El juego de la reflexión. Trate de rodear y capturar los cuadrados de su oponente sin dejar los suyos al descubierto. Para un jugador contra el ordenador. Sólo teclado.

```
10 'AMTHELLO, por M. J. Gribbins
20 'copyright (c) AMSOFT 1984
30 '
40 BORDER 14
50 CLEAR
60 MODE 1:PEN 0:PAPER 1:CLS
70 INK 0,0:INK 1,14:INK 2,18:INK 3,26
80 LOCATE 2,3:PEN 3:PRINT"A":LOCATE 3,4:PRINT"M":LOCATE 4,5:P
   RINT"T":LOCATE 5,6:PRINT"H"
90 LOCATE 6,7:PRINT"E":LOCATE 7,8:PRINT"L":LOCATE 8,9:PRINT"L
   ":LOCATE 9,10:PRINT"O"
100 WINDOW #1,2,39,22,25:PAPER #1,1:PEN #1,0:CLS#1
110 PEN 0
120 LOCATE #1,5,1:PRINT#1,"LAS NEGRAS SIEMPRE EMPIEZAN"
130 LOCATE #1,4,3:PRINT#1,"ELIJA BLANCAS <B> O NEGRAS <N>"
140 b$=INKEY$:IF b$="" THEN 140
150 IF b$="B" OR b$="b" THEN q%=3:n%=0:GOTO 210
160 IF b$="N" OR b$="n" THEN q%=0:n%=3:GOTO 210
170 CLS#1:LOCATE #1,4,3
180 PRINT#1,"    SOLO BLANCAS O NEGRAS"
190 FOR t=0 TO 1000:NEXT t
200 GOTO 140
210 DIM c%(10,10),p%(9,9),c1%(8),c2%(8),cx%(9),cy%(9)
220 i1%=2:j1%=2:i2%=7:j2%=7
230 FOR i%=0 TO 9
240 c%(i%,o%)=6:c%(0,i%)=6
250 c%(9,i%)=6:c%(i%,9)=6
260 NEXT i%
270 FOR i%=1 TO 8
280 READ c1%(i%),c2%(i%)
290 FOR j%=1 TO 8
300 READ p%(i%,j%)
310 c%(i%,j%)=6
320 NEXT j%:NEXT i%
330 c%(4,4)=3:c%(4,5)=0:c%(5,4)=0:c%(5,5)=3
340 FOR k%=1 TO 58
350 READ ar%,br%,cr%,dr%
360 PLOT ar%,br%:DRAW cr%,dr%,0
370 NEXT k%
```

```
380 GOSUB 1460
390 IF q%=3 GOTO 770
400 CLS#1:INPUT #1," ELIJA FILA ";e%
410 IF e%<1 OR e%>8 GOTO 400
420 LOCATE #1,1,3:INPUT #1," ELIJA COLUMNA ";d%
430 IF d%<1 OR d%>8 GOTO 420
440 IF c%(d%,e%)=6 GOTO 480
450 CLS#1:LOCATE #1,5,2:PRINT#1,"ESE CUADRADO YA ESTA OCUPADO
    !"
460 FOR t=1 TO 1000:NEXT t
470 GOTO 400
480 PLOT 270+(30*d%),70+(30*e%):DRAW 290+(30*d%),89+(30*e%),q
    %
490 PLOT 290+(30*d%),70+(30*e%):DRAW 270+(30*d%),89+(30*e%),q
    %
500 GOTO 540
510 FOR m%=0 TO 19 STEP 2:PLOT 270+(30*d%),70+m%+(30*e%)
520 DRAW 290+(30*d%),70+m%+(30*e%),6:NEXT m%
530 GOTO 400
540 vrx%=0
550 FOR k%=1 TO 8
560 vr%=0:c3%=d%:c4%=e%
570 c3%=c3%+c1%(k%):c4%=c4%+c2%(k%)
580 IF c%(c3%,c4%)=n% GOTO 590 ELSE 600
590 vr%=vr%+1:GOTO 570
600 IF c%(c3%,c4%)=6 GOTO 610 ELSE 620
610 NEXT k%:GOTO 670
620 IF vr%=0 GOTO 610 ELSE 630
630 vrx%=vrx%+vr%
640 c3%=c3%-c1%(k%):c4%=c4%-c2%(k%)
650 IF c%(c3%,c4%)=6 GOTO 610 ELSE 660
660 c%(c3%,c4%)=q%:GOTO 640
670 IF vrx%=0 GOTO 680 ELSE 710
680 CLS#1:PRINT#1,"ESO NO ES POSIBLE"
690 FOR t=1 TO 1000:NEXT t
700 GOTO 510
710 e%=e%:d%=d%:vrx%=vrx%
720 CLS#1:PRINT#1,"USTED VA A LA LINEA ";e%
730 PRINT#1,"      Y A LA COLUMNA ";d%
740 LOCATE #1,2,4:PRINT#1,"CON LO QUE CONSIGUE ";vrx%;" CUADR
    ADO(S)"
750 c%(d%,e%)=q%:GOSUB 1710
760 GOSUB 1460
770 CLS#1:LOCATE #1,10,2:PRINT#1,"AHORA ME TOCA A MI ...!"
780 p%=0:vrx%=0:vry%=0
```

```
790 IF i1%*j1%=1 AND i2%*j2%=64 GOTO 860
800 FOR k%=2 TO 7
810 IF c%(2,k%)<>6 THEN i1%=1
820 IF c%(7,k%)<>6 THEN i2%=8
830 IF c%(k%,2)<>6 THEN j1%=1
840 IF c%(k%,7)<>6 THEN j2%=8
850 NEXT k%
860 FOR i%=i1% TO i2%
870 FOR j%=j1% TO j2%
880 IF c%(i%,j%)=6 GOTO 1030
890 NEXT j%:NEXT i%
900 IF p%>0 THEN 1000
910 IF pas%=1 GOTO 920 ELSE 940
920 CLS#1:PRINT#1,"BLOQUEADO! YO TAMBIEN TENGO QUE PASAR. FIN
    DE LA PARTIDA"
930 FOR t=1 TO 1000:NEXT t:GOTO 1550
940 CLS#1:LOCATE #1,18,2:PRINT#1,"TENGO QUE PASAR"
950 GOSUB 2720
960 IF pas%=1 GOTO 970 ELSE 990
970 CLS#1:PRINT#1,"BLOQUEADO! USTED TAMBIEN TIENE QUE PASAR.
    FIN DE LA  PARTIDA."
975 RETURN
980 FOR t=1 TO 1000:NEXT t:GOTO 1550
990 GOTO 400
1000 IF lc%=0 THEN lc%=1:RANDOMIZE lc%:rl%=RND(lc%)
1010 cx1%=cx%(rl%):cx2%=cy%(rl%)
1020 GOTO 1220
1030 vrx%=0
1040 FOR k%=1 TO 8
1050 vr%=0:c3%=i%:c4%=j%
1060 c3%=c3%+c1%(k%):c4%=c4%+c2%(k%)
1070 IF c%(c3%,c4%)=q% GOTO 1080 ELSE 1090
1080 vr%=vr%+1:GOTO 1060
1090 IF c%(c3%,c4%)=6 GOTO 1100 ELSE 1110
1100 NEXT k%:GOTO 1130
1110 IF vr%=o% GOTO 1100 ELSE 1120
1120 vrx%=vrx%+vr%:GOTO 1100
1130 IF vrx%=0 GOTO 890
1140 IF p%(i%,j%)<p% GOTO 890
1150 IF p%(i%,j%)>p% GOTO 1160 ELSE 1170
1160 p%=p%(i%,j%):vry%=vrx%:lc%=0:cx%(0)=i%:cy%(0)=j%:GOTO 89
     0
1170 IF vry%>vrx% GOTO 890
1180 IF vry%<vrx% GOTO 1190 ELSE 1200
1190 lc%=0:vry%=vrx%:cx%(0)=i%:cy%(0)=j%:GOTO 890
```

```
1200 lc%=lc%+1:cx%(lc%)=i%:cy%(lc%)=j%
1210 GOTO 890
1220 cx2%=cx2%:cx1%=cx1%:vry%=vry%
1230 CLS#1:PRINT#1,"YO VOY A LA FILA ";cx2%
1240 PRINT#1,"  Y A LA COLUMNA ";cx1%
1250 LOCATE #1,1,4:PRINT#1,"CON LO QUE CONSIGO ";vry%;" CUADR
     ADO(S)"
1260 PLOT 270+(30*cx1%),70+(30*cx2%):DRAW 290+(30*cx1%),89+(3
     0*cx2%),n%
1270 PLOT 290+(30*cx1%),70+(30*cx2%):DRAW 270+(30*cx1%),89+(3
     0*cx2%),n%
1280 FOR t=1 TO 1000:NEXT t
1290 FOR k%=1 TO 8
1300 vr%=0:c3%=cx1%:c4%=cx2%
1310 c3%=c3%+c1%(k%):c4%=c4%+c2%(k%)
1320 IF c%(c3%,c4%)=q% GOTO 1330 ELSE 1340
1330 vr%=vr%+1:GOTO 1310
1340 IF c%(c3%,c4%)=6 GOTO 1350 ELSE 1360
1350 NEXT k%:GOTO 1400
1360 IF vr%=0 GOTO 1350
1370 c3%=c3%-c1%(k%):c4%=c4%-c2%(k%)
1380 IF c%(c3%,c4%)=6 GOTO 1350
1390 c%(c3%,c4%)=n%:GOTO 1370
1400 c%(cx1%,cx2%)=n%
1410 GOSUB 2720
1420 GOSUB 1460
1430 IF pas%=1 GOTO 1440 ELSE 1450
1440 CLS#1:PRINT#1,"    USTED TIENE QUE PASAR":FOR t=1 TO 100
     0:NEXT t:GOTO 770
1450 GOTO 400
1460 FOR i%=1 TO 8
1470 FOR j%=1 TO 8
1480 FOR m%=1 TO 19 STEP 2
1490 z%=270+(30*i%):h%=70+(30*j%):w%=h%+m%
1500 PLOT z%,w%:DRAW z%+20,w%,c%(i%,j%)
1510 NEXT m%:NEXT j%:NEXT i%
1520 x%=x%+1
1530 IF x%=61 GOTO 1550
1540 RETURN
1550 cq%=0:cn%=0
1560 FOR i%=1 TO 8
1570 FOR j%=1 TO 8
1580 IF c%(i%,j%)=q% THEN cq%=cq%+1
1590 IF c%(i%,j%)=n% THEN cn%=cn%+1
1600 NEXT j%:NEXT i%
```

```
1610 IF cq%>cn% GOTO 1680
1620 IF cq%=cn% GOTO 1630 ELSE 1650
1630 CLS#1:LOCATE #1,25,2:PRINT#1,"BLOQUEADO"
1640 END
1650 CLS#1:LOCATE #1,5,1:PRINT#1,"USTED TIENE ";cq%;" CUADRAD
     OS. YO TENGO ";cn%
1660 LOCATE #1,11,3:PRINT#1,"YO HE GANADO ... !!!"
1670 END
1680 CLS#1:LOCATE #1,5,1:PRINT#1,"USTED TIENE ";cq%;" CUADRAD
     OS. YO TENGO ";cn%
1690 LOCATE #1,5,3:PRINT#1,"MUY BIEN. USTED HA GANADO ... !!!
     "
1700 END
1710 IF c%(2,2)=q% AND (c%(3,1)=n% OR c%(1,3)=n%) GOTO 1720 E
     LSE 1730
1720 p%(3,1)=1:p%(1,3)=1
1730 IF c%(7,7)=q% AND (c%(8,6)=n% OR c%(6,8)=n%) GOTO 1740 E
     LSE 1750
1740 p%(8,6)=1:p%(6,8)=1
1750 IF c%(2,7)=q% AND (c%(1,6)=n% OR c%(3,8)=n%) GOTO 1760 E
     LSE 1770
1760 p%(1,6)=1:p%(3,8)=1
1770 IF c%(7,2)=q% AND (c%(6,1)=n% OR c%(8,3)=n%) GOTO 1780 E
     LSE 1790
1780 p%(6,1)=1:p%(8,3)=1
1790 IF d%=1 OR d%=8 OR e%=1 OR e%=8 GOTO 1820
1800 IF cx1%=1 OR cx1%=8 OR cx2%=1 OR cx2%=8 GOTO 1820
1810 RETURN
1820 FOR j%=1 TO 8 STEP 7
1830 FOR i%=2 TO 7
1840 IF c%(i%,j%)=n% GOTO 1850 ELSE 1860
1850 p%(i%+1,j%)=21:p%(i%-1,j%)=21
1860 IF c%(j%,i%)=n% GOTO 1870 ELSE 1880
1870 p%(j%,i%+1)=21:p%(j%,i%-1)=21
1880 NEXT i%
1890 FOR i%=2 TO 7
1900 IF c%(i%,j%)=q% GOTO 1910 ELSE 1920
1910 p%(i%+1,j%)=2:p%(i%-1,j%)=2
1920 IF c%(j%,i%)=q% GOTO 1930 ELSE 1940
1930 p%(j%,i%+1)=2:p%(j%,i%-1)=2
1940 NEXT i%:NEXT j%
1950 p%(1,2)=1:p%(1,7)=1:p%(2,1)=1:p%(7,1)=1
1960 p%(2,8)=1:p%(7,8)=1:p%(8,2)=1:p%(8,7)=1
1970 FOR i%=2 TO 7
1980 IF c%(1,i%-1)=q% AND c%(1,i%+1)=q% THEN p%(1,i%)=25
```

```
1990 IF c%(8,i%-1)=q% AND c%(8,i%+1)=q% THEN p%(8,i%)=25
2000 IF c%(i%-1,1)=q% AND c%(i%+1,1)=q% THEN p%(i%,1)=25
2010 IF c%(i%-1,8)=q% AND c%(i%+1,8)=q% THEN p%(i%,8)=25
2020 NEXT i%
2030 FOR j%=1 TO 8 STEP 7
2040 FOR i%=4 TO 8
2050 IF c%(j%,i%)<>n% GOTO 2140
2060 ic%=i%-1:IF c%(j%,ic%)=6 GOTO 2140
2070 IF c%(j%,ic%)=q% GOTO 2080 ELSE 2090
2080 ic%=ic%-1:GOTO 2070
2090 IF c%(j%,ic%)=6 GOTO 2110
2100 GOTO 2140
2110 IF ic%=0 GOTO 2140
2120 IF c%(j%,i%+1)=q% AND c%(j%,ic%-1)=6 GOTO 2140
2130 p%(j%,ic%)=26
2140 IF c%(i%,j%)<>n% GOTO 2230
2150 ic%=i%-1:IF c%(ic%,j%)=6 GOTO 2230
2160 IF c%(ic%,j%)=q% GOTO 2170 ELSE 2180
2170 ic%=ic%-1:GOTO 2160
2180 IF c%(ic%,j%)=6 GOTO 2200
2190 GOTO 2230
2200 IF ic%=0 GOTO 2230
2210 IF c%(i%+1,j%)=q% AND c%(ic%-1,j%)=6 GOTO 2230
2220 p%(ic%,j%)=26
2230 NEXT i%
2240 FOR i%=1 TO 5
2250 IF c%(j%,i%)<>n% GOTO 2340
2260 ic%=i%+1:IF c%(j%,ic%)=6 GOTO 2340
2270 IF c%(j%,ic%)=q% GOTO 2280 ELSE 2290
2280 ic%=ic%+1:GOTO 2270
2290 IF c%(j%,ic%)=6 GOTO 2310
2300 GOTO 2340
2310 IF ic%=9 GOTO 2340
2320 IF c%(j%,i%-1)=q% AND c%(j%,ic%+1)=6 GOTO 2340
2330 p%(j%,ic%)=26
2340 IF c%(i%,j%)<>n% GOTO 2430
2350 ic%=i%+1:IF c%(ic%,j%)=6 GOTO 2430
2360 IF c%(ic%,j%)=q% GOTO 2370 ELSE 2380
2370 ic%=ic%+1:GOTO 2360
2380 IF c%(ic%,j%)=6 GOTO 2400
2390 GOTO 2430
2400 IF ic%=9 GOTO 2430
2410 IF c%(i%-1,j%)=q% AND c%(ic%+1,j%)=6 GOTO 2430
2420 p%(ic%,j%)=26
2430 NEXT i%:NEXT j%
```

```
2440 IF c%(1,1)=n% GOTO 2450 ELSE 2460
2450 FOR i%=2 TO 6:p%(1,i%)=20:p%(i%,1)=20:NEXT i%
2460 IF c%(1,8)=n% GOTO 2470 ELSE 2480
2470 FOR i%=2 TO 6:p%(i%,8)=20:p%(1,9-i%)=20:NEXT i%
2480 IF c%(8,1)=n% GOTO 2490 ELSE 2500
2490 FOR i%=2 TO 6:p%(9-i%,1)=20:p%(8,i%)=20:NEXT i%
2500 IF c%(8,8)=n% GOTO 2510 ELSE 2520
2510 FOR i%=3 TO 7:p%(i%,8)=20:p%(8,i%)=20:NEXT i%
2520 IF c%(1,1)<>6 THEN p%(2,2)=5
2530 IF c%(1,8)<>6 THEN p%(2,7)=5
2540 IF c%(8,1)<>6 THEN p%(7,2)=5
2550 IF c%(8,8)<>6 THEN p%(7,7)=5
2560 p%(1,1)=30:p%(1,8)=30:p%(8,1)=30:p%(8,8)=30
2570 FOR i%=3 TO 6
2580 IF c%(1,i%)=n% THEN p%(2,i%)=4
2590 IF c%(8,i%)=n% THEN p%(7,i%)=4
2600 IF c%(i%,1)=n% THEN p%(i%,2)=4
2610 IF c%(i%,8)=n% THEN p%(i%,7)=4
2620 NEXT i%
2630 IF c%(7,1)=q% AND c%(4,1)=n% AND c%(6,1)=6 AND c%(5,1)=6
     THEN p%(6,1)=26
2640 IF c%(1,7)=q% AND c%(1,4)=n% AND c%(1,6)=6 AND c%(1,5)=6
     THEN p%(1,6)=26
2650 IF c%(2,1)=q% AND c%(5,1)=n% AND c%(3,1)=6 AND c%(4,1)=6
     THEN p%(3,1)=26
2660 IF c%(1,2)=q% AND c%(1,5)=n% AND c%(1,3)=6 AND c%(1,4)=6
     THEN p%(1,3)=26
2670 IF c%(8,2)=q% AND c%(8,5)=n% AND c%(8,3)=6 AND c%(8,4)=6
     THEN p%(8,3)=26
2680 IF c%(2,8)=q% AND c%(5,8)=n% AND c%(3,8)=6 AND c%(4,8)=6
     THEN p%(3,8)=26
2690 IF c%(8,7)=q% AND c%(8,4)=n% AND c%(8,5)=6 AND c%(8,6)=6
     THEN p%(8,6)=26
2700 IF c%(7,8)=q% AND c%(4,8)=n% AND c%(5,8)=6 AND c%(6,8)=6
     THEN p%(6,8)=26
2710 RETURN
2720 pas%=0
2730 FOR i%=1 TO 8
2740 FOR j%=1 TO 8
2750 IF c%(i%,j%)=q% GOTO 2780
2760 NEXT j%:NEXT i%
2770 pas%=1:RETURN
2780 FOR k%=1 TO 8
2790 vr%=0:c3%=i%:c4%=j%
2800 c3%=c3%+c1%(k%):c4%=c4%+c2%(k%)
```

```
2810 IF c3%<1 OR c3%>8 GOTO 2820 ELSE 2830
2820 NEXT k%:GOTO 2760
2830 IF c4%<1 OR c4%>8 GOTO 2820 ELSE 2840
2840 IF c%(c3%,c4%)=n% GOTO 2850 ELSE 2860
2850 vr%=vr%+1:GOTO 2800
2860 IF c%(c3%,c4%)=q% GOTO 2820 ELSE 2870
2870 IF vr%>0 THEN RETURN
2880 GOTO 2820
2890 DATA 1,0,30,1,20,10,10,20,1,30,1,1,1,1,3
2900 DATA 3,3,3,1,1,0,1,20,3,5,5,5,5,3,20,-1,1,10,3,5
2910 DATA 0,0,5,3,10,-1,0,10,3,5,0,0,5,3,10,-1
2920 DATA -1,20,3,5,5,5,5,3,20,0,-1,1,1,3,3,3,3,1,1,1,-1,30,1
     ,20,10,10,20,1,30
2930 DATA 263,100,263,120,270,130,255,130,255,130,255,140,255
     ,140,270,140
2940 DATA 270,140,270,150,270,150,255,150,255,160,270,160,270
     ,160,270,180
2950 DATA 270,180,255,180,270,170,255,170,270,190,270,210,270
     ,200,255,200
2960 DATA 255,200,255,210,255,220,270,220,270,220,270,230,270
     ,230,255,230
2970 DATA 255,230,255,240,255,240,270,240,255,250,270,250,270
     ,250,270,260
2980 DATA 270,260,255,260,255,250,255,270,270,280,270,300,270
     ,300,255,300
2990 DATA 255,310,255,330,255,330,270,330,270,330,270,310,270
     ,310,255,310
3000 DATA 255,320,270,320
3010 DATA 310,355,310,375,350,355,335,355,335,355,335,365,335
     ,365,350,365
3020 DATA 350,365,350,375,350,375,335,375,365,355,380,355,380
     ,355,380,375
3030 DATA 380,375,365,375,380,365,365,365,410,355,410,375,410
     ,365,395,365
3040 DATA 395,365,395,375,425,355,440,355,440,355,440,365,440
     ,365,425,365
3050 DATA 425,365,425,375,425,375,440,375,455,375,455,355,455
     ,355,470,355
3060 DATA 470,355,470,365,470,365,455,365,485,375,500,375,500
     ,375,500,355
3070 DATA 515,375,515,355,515,355,530,355,530,355,530,375,530
     ,375,515,375
3080 DATA 515,365,530,365
```

Apéndice 3
Indice

(Las referencias corresponden a los números de capítulo y de página; por ejemplo, 1.42 significa capítulo 1 página 42.)

A

ABS 3.3
AFTER 3.3 5.25
Alimentación, interruptor 1.4 1.5
Altavoces externos 1.8 1.54
Ampliaciones del sistema residentes 4.41
Amplificador externo 1.8 4.37
AND 3.4 5.15
ASC 3.5
ASCII 4.8 4.21
ASCII, caracteres 4.8 4.9 5.11
ASCII, ficheros 3.61 4.30
ATN 3.5
Auriculares 1.8 1.54
AUTO 2.8 3.5

B

BASIC 1.20 3.1 4.28 5.5
BIN$ 3.6
bit 5.7
BORDER 1.39 3.6 4.5
BORR, tecla 1.10
Borrado de los canales de sonido 3.73 5.32
BREAK 3.6
BRILLO, control 1.4
Byte 5.7

C

CALL 3.7
Capacidad de los discos 1.41
Caracteres 1.46 4.9 5.11
Caracteres de control 4.3
Caracteres definibles por el usuario 3.68 5.17
Caracteres expansibles 3.32 4.22
Caracteres transparentes 4.4
CHAIN 3.7
CHAIN MERGE 3.8
CHR$ 1.45 3.8 5.12
CINT 3.9
Circunferencias 1.49
CLEAR 3.9
CLG 3.9

CLOSEIN 2.9 3.10
CLOSEOUT 2.8 3.10
CLR, tecla 1.12
CLS 1.20 3.10 4.3
Código de máquina 4.6
Códigos de control 4.2 4.3
Colas de sonido 3.45 3.66 4.7 5.31 5.37
Colores 1.38 1.41
Colores parpadeantes 1.42 3.64
Conexión de una clavija de red 1.1
Conexión del ordenador 1.2
Conexión de periféricos 1.7 4.36 4.37 4.38
CONT 3.11 4.29
CONTRASTE, control 1.4
COPIA, tecla 1.26
COS 3.11
CREAL 3.11
Cursor, teclas del 1.9

D

DATA 3.12 4.28 5.26
DEF FN 3.12 4.30
DEFINT 3.13
DEFREAL 3.13
DEFSTR 3.14
DEG 3.14
DELETE 3.14
DI 3.15 5.26
DIM 2.3 3.16 4.29
DRAW 1.48 3.16
DRAWR 3.17

E

Edición de programas 1.25
EDIT 1.25 3.17
EI 3.17 5.26
ELSE 1.27 3.18
ENC., piloto 1.4 1.5
END 3.18
ENT 1.58 3.18 5.35
ENV 1.56 3.20 5.32
Envolvente de tono 1.58 3.18 5.35
Envolvente de volumen 1.56 3.20 5.32
EOF 3.22 4.30
ERASE 3.22
ERL 3.22
ERR 3.23
ERROR 3.23
ESC, tecla 1.12
Escritura transparente 4.4

Estéreo, zócalo 4.37
Estereofonía 1.8 1.54
EVERY 3.24 5.26
EXP 3.24
E/S 4.36 4.44
E/S, zócalo 1.8 1.54

F Ficheros ASCII 3.61 4.30
Ficheros de BASIC 3.61
Ficheros binarios 3.61
Ficheros protegidos 3.61
FIJA MAYS, tecla 1.10
FIX 3.24
FN 3.25
FOR 1.28 3.25 4.28 4.30 5.12
FRE 3.25

G GOSUB 1.29 3.26
GOTO 1.22 3.26
Grabación de variables 2.9 3.78
Gráficos 1.45

H Hardware 4.42
HEX$ 3.26
HIMEM 3.26 4.42

I IF 1.27 3.27
Impresoras 1.7 4.39
INK 1.39 3.27 4.5
INKEY 3.28 4.40
INKEY$ 2.11 3.29 4.40
INP 3.29
INPUT 1.23 2.2 3.29 4.29
INSTR 2.4 3.30
INT 3.31
Interfaz serie 5.3
Interrupciones 4.7 5.24
INTRO, tecla 1.9

J JOY 3.31 4.40
Joysticks 1.7 4.21 4.23 4.40
JOYSTICK, zócalo 1.7 4.36

K KEY 3.32 4.22
KEY DEF 3.32 4.22 4.40

L

LEFT$ 3.33
LEN 2.8 3.33
LET 3.34
LINE INPUT 3.34
LIST 1.22 1.44 3.34
Listas 2.3 3.16 3.22 4.29
LOAD 3.35
LOCATE 1.45 3.35 4.6
LOG 3.36
LOG10 ..... 3.36
Lógica 3.4 3.41 3.46 3.79 5.14
LOWER$ 3.36

M

Matrices 2.3 3.16
MAX 3.37
MAYS, teclas 1.10
MEMORY 3.37 4.28 4.42
Mensajes de error 4.28
Menú 2.5 3.44
MERGE 3.37
MID$ 3.38
MIN 3.39
MOD 1.32 3.39
MODE 1.38 3.39 4.3
Modos de tinta 4.5
Modulador/fuente de alimentación (MP1) 1.3 1.5 1.38 1.54
Monitor 1.2
MONITOR, zócalo 1.2 1.3 4.37
MOVE 1.49 3.40
MOVER 3.40

N

NEW 1.50 3.40
NEXT 1.28 3.41 4.28 4.30 5.12
NOT 3.41 5.14
Notas musicales 4.24
Números aleatorios 3.59
Números binarios 5.6
Números de los errores 4.28
Números hexadecimales 5.9

O

ON BREAK GOSUB 3.42
ON BREAK STOP 3.42
ON ERROR GOTO 3.43 4.30
ON GOSUB 2.5 3.43
ON GOTO 3.44
ON SQ GOSUB 3.44 4.7 5.38
OPENIN 2.9 3.45 4.30

OPENOUT 2.8 3.45 4.30
Operadores 1.30 5.15
OR 3.46 5.15
ORIGIN 1.51 3.46
OUT 3.47

P

Palabras clave 1.20 3.1
PAPER 1.39 3.47 4.4
PEEK 3.47
PEN 1.39 3.48 4.4
Periféricos 1.7 5.2
PI 3.48
Piloto (de la unidad de disco) 1.8
Plantillas 4.32 4.33 4.34 4.35
PLOT 1.47 3.48
PLOTR 3.49
POKE 3.49
Potenciación 1.33 1.35
PRINT 1.21 3.50 5.18
PRINT SPC 3.51 5.19
PRINT TAB 3.51 5.19
PRINT USING 3.51 5.20
PUERTA DEL USUARIO, zócalo 1.7 4.36

R

RAD 3.53
Raíz cuadrada 1.33 3.66
Raíz cúbica 1.33
RANDOMIZE 3.54
READ 3.54 7.28 5.26
RELEASE 3.55 5.36
REM 1.28 2.2 3.55
REMAIN 3.56 5.26
RENUM 2.7 3.56
RESTORE 3.57 5.28
RESUME 3.57 4.30
RESUME NEXT 3.58
Retención de los canales de sonido 3.62 5.32
RETURN 1.29 3.58 7.28
RETURN, tecla 1.9 1.12
RIGHT$ 3.59
RND 3.59
ROM adicional 4.44
ROUND 3.60
RS232C 5.3
RSX 4.41
RUN 1.21 3.60

S SAVE ... 3.61
SGN ... 3.61
SIN ... 3.62
Sincronización de canales ... 3.62 5.31
Sintetizador de voz ... 1.8 5.3
Software ... 1.14
SOUND ... 1.54 3.62 4.24 5.29
SPACE$ ... 3.64
SPC ... 3.64 5.19
SPEED INK ... 3.64
SPEED KEY ... 3.65
SPEED WRITE ... 3.65
SQ ... 3.66 5.38
SQR ... 1.33 3.66
STEP ... 1.28 3.66
STOP ... 3.67
STR$ ... 3.67
STRING$ ... 3.67
Suma de verificación ... 5.27
SWAP ... 3.68
SYMBOL ... 3.68 4.5 5.17
SYMBOL AFTER ... 3.70 4.42
Syntax error ... 1.13 7.28

T TAB ... 3.70 5.19
TAG ... 3.70
TAGOFF ... 3.71
TAN ... 3.71
Teclado ... 1.9 4.21 4.22 4.23 4.40
Teclas definibles por el usuario ... 3.32 4.21 4.22 4.23
Televisor ... 1.3 1.5 1.54
TEST ... 3.72
TESTR ... 3.72
THEN ... 1.27 3.72
TIME ... 3.73
TO ... 3.73
TROFF ... 3.73
TRON ... 3.73

U UNIDAD DE DISCO, zócalo ... 1.8 4.38
UNT ... 3.74
UPPER$ ... 3.74
USING ... 3.74 5.20

V VAL ... 3.75
Variables ... 1.24 4.31 5.12
Variables, grabación de ... 2.8 3.61 3.78
Variables literales ... 1.24 4.29

Vibrato 5.34
Volcado de pantalla 3.61
VOLUMEN, control 1.8 1.54
VPOS 3.75

W WAIT 3.75
WEND 3.76 4.31
WHILE 3.76 4.28 4.31
WIDTH 3.76
WINDOW 2.10 3.77 4.5 5.22
WINDOW SWAP 3.77 5.23
WRITE 2.8 3.78 5.19

X XOR 3.79 5.17
XPOS 3.79

Y YPOS 3.80

Z Zócalos de corriente contínua 1.2 1.3
ZONE 3.80 5.18

GRUPO INDESCOMP AMSTRAD ESPAÑA

Aravaca, 22. 28040 MADRID. Tel. 459 30 01, Télex 47660 INSC E, Fax 459 22 92
Delegación en Cataluña: Tarragona, 110. 08015 BARCELONA, Tel. 325 10 58

EL MATERIAL EXHIBIDO
ES SOLO PARA USO
EDUCATIVO, NO COMERCIAL

ENCICLOPEDIA DE
REFERENCIA | EDUCATIVA

RETRO TECNIA

PARA ENTENDER LA EVOLUCION DE LOS ORDENADORES A TRAVES DEL TIEMPO